OPTIMIZATION OF CONTROL DEVICE LOCATIONS AND SIZES IN MINE VENTILATION SYSTEMS

by

Xing Wu

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute & State University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY
in
Mining Engineering

APPROVED:

[illegible]

E. Topuz, Chairman

[illegible]

C. Haycocks

Michael Lucas

J.R. Lucas

[illegible]

M. Karfakis

[illegible]

M. McPherson

September, 1992

Blacksburg, Virginia

c.2

LD
5655
V856
1992
W84
c.2

OPTIMIZATION OF CONTROL DEVICE LOCATIONS AND SIZES IN MINE VENTILATION SYSTEMS

by

Xing Wu

Committee Chairman: Ertugrul Topuz

Mining and Minerals Engineering

(ABSTRACT)

This study proposes an improved methodology for solving the semi–controlled ventilation network problem. After analysis and consideration of the objective and legal, technical and operation convenience factors, the semi–controlled network problem is formulated through a nonlinear nonconvex programming model. Using the special ordered sets variables, the nonconvex problem is linearized and then optimized by the modified branch and bound procedure where the automatic interpolation technique is used to improve the accuracy to which the each nonlinear function is approximated. The global optimality of the methodology is on the computational theory basis. And the applicability of the methodology to the generalized ventilation problem is investigated, and it is demonstrated with a number of examples.

This study also compares the several methods which have been used so far to optimize the underground ventilation networks.

DEDICATION

This dissertation is dedicated to my parents for their constant encouragement, support, and love during my education.

ACKNOWLEDGEMENTS

I would like to thank my dissertation advisor, Dr. E. Topuz for his guidance and support during the course of my research. I also thank my committee members, Dr. M. McPherson, Dr. C. Haycocks, Dr. M. Karfakis, and Dr. J. R. Lucas for their invaluable advice and criticism. I also appreciate valuable advice received from Dr. Y.J. Wang during the early stage of my study.

Special appreciation is extended to Dr. Walker and Mrs. Walker for their considerable help, support, and review of this manuscript.

I extend much appreciation to Miss Irene Lu for her understanding and support during the darkest hours of this project, and to my daughter, Sherry Wu, for the many patient hours spent amusing herself while I worked.

I would also like to thank my colleagues and friends for their assistance and encouragement during my study.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER I INTRODUCTION

1.1 Introduction to the Problem

The objective of this dissertation is to determine the location and size of all fans and regulators, in order to minimize the overall cost of ventilation and satisfy all physical, legal, and operational constraints. It includes the formulation of the problem, the development of a computational theory, a solution methodology, and a computer program.

The research presented in this dissertation seeks to solve the ventilation network problem. The methodology developed can be used to solve natural splitting, controlled splitting, and semi–controlled splitting network problems. Consequently, it can be used for both ventilation network design and network analysis.

The research compares several methods presently used in the optimization of size and location of control devices in underground mine ventilation systems. The purpose is to identify their advantages, weaknesses, and limitations and to develop an improved methodology.

1.2 Problem Statement

Excavation in the earth under even normal circumstances can be fraught with environmental problems and hazards. In underground mining, the most critical aspect of the environment is to control the atmosphere of the work place. Among the set of atmospheric control alternatives, mine ventilation is the measure most commonly used. Mine ventilation is carried out by mine ventilation systems consisting of airways and ventilation control devices. The system provides the required quantity and quality of air flow at three to ten percent of the total mining cost.

The air flow within an underground network in mine ventilation can be categorized in three ways: natural splitting, controlled splitting, and semi–controlled splitting. In natural splitting, the air flow in branches of a ventilation network apportions itself according to the aerodynamic resistance of the branches. In controlled splitting, a prescribed quantity of air is circulated through each branch. In semi–controlled splitting, flow in some branches is fixed to a desired quantity while in other branches it splits naturally.

Optimization of the location of ventilation devices and their sizing is an objective in ventilation design. Due to rapid advances in mathematical programming and computers in the past three decades, several design methods have been established. These methods can be classified into two groups: (1) those that deal with network problems of controlled flow and (2) those designated for the solution of network problems of semi–controlled splitting. Those in the first group are pursued with the application of linear programming, CPM together with cutset operation, the out–of–kilter method, the network simplex method, and critical path crashing. Those included in group two are sought with the implementation of the

Hardy Cross iteration method, the flow–path identification technique, the transformation of a semi–controlled network to a controlled network, simulation, the nonlinear equation algorithm and the nonlinear network optimization algorithm. Theoretically, solving the semi–controlled network problem is more difficult than solving the controlled network problem because the semi–controlled network requires simultaneous determination of ventilation device location and size and flow distribution. The present techniques for the solution of the semi–controlled splitting network problem combine existing algorithms.

Ventilation network analysis and design are two different tasks. For a given ventilation network, analysis enables the determination of iteratively balanced air flow rates to satisfy Kirchhoff's laws. On the other hand, design determines the locations and sizes of ventilation control devices for a given layout of a network. However, analysis and design are interrelated and complementary. A ventilation network analyzer can be used to evaluate a ventilation layout and provide the planner with the information needed for the modification of the layout. As a result, ventilation engineers are able to modify the layout based on the results from the analyzer. Ideally, however, a comprehensive tool should combine the analysis of an existing network with the design of a new one.

In recent years, requirements for air flow quantity and quality have increased because of deeper mines, more rigorous environmental standards, and the use of high productivity equipment. The cost of mine ventilation has risen accordingly, and, as a result, optimization of device location and size in mining ventilation systems has become more critical.

Booster fans have been used in underground mines throughout the world. Both theoretically and practically, it has been proven that proper use of booster fans in underground ventilation systems can improve flow distribution and reduce power consumption. This has raised a new challenge in ventilation system design: determination of the optimal combination of main fan and booster fans in a mine ventilation system.

In this study, a solution for the generalized ventilation network problem is attempted through the application of a nonlinear nonconvex programming algorithm. The method formulates the ventilation network problem as a nonlinear programming model and uses techniques of special ordered sets (Beale and Tomlin, 1970), column generation (Lasdon, 1970) and branch–and–bound (Hillier and Lieberman, 1986) to obtain the optimal solution. The objective is to overcome the drawbacks of the existing techniques, while insuring the optimum on theoretical grounds.

1.3 Dissertation Organization

Chapter II states the ventilation network problem. Chapter III reviews the existing methods for optimization of location and size of ventilation devices in underground ventilation networks. Chapter IV presents the fundamental concepts and methodology for solving the nonlinear, nonconvex programming problem. Chapter V describes the solution to the problem. The Chapter VI presents applications of the methodology. Chapter VII states conclusions and recommendations for future research.

CHAPTER II PROBLEM STATEMENT

2.1 Introduction

The problem of optimization of ventilation control device locations and sizes in an underground mine ventilation system can be stated as follows: for a given ventilation network, which means that the topology of the ventilation network is known and fixed, to determine optimal locations and sizes for all fans and regulators in the system. The optimal solution minimizes the total cost of ventilation and satisfies all legal and technical requirements.

The solution is rather restricted. First, the topology of the ventilation network is known and fixed i.e., size and arrangement of air ways is not involved in optimization. Second, the solution only corresponds to one fixed layout, in other words, it refers to one point of the network development.

2.2 Notation

The following notation is used throughout the discussion of this dissertation.

Q_j = the flow rate through branch j (m^3/s,cubic meter per second)

q_j = the fixed flow rate through branch j (m^3/s)

HF_j = the fan pressure in branch j (Pa, Pascal)

HR_j = the pressure loss for the regulator in branch j (Pa)

HL_j = the pressure loss for branch j (Pa)

HN_j = the natural ventilation pressure across branch j (Pa)

R_j = resistance factor for branch j ($N \cdot s^2/m^8$)

L_j = the lower bound of the flow through branch j (m^3/s)

U_j = the upper bound of the flow through branch j (m^3/s)

HF_{jmin} = the minimum fan pressure in branch j (Pa)

HF_{jmax} = the maximum fan pressure in branch j (Pa)

n = number of nodes in network

b = number of branches in network

m = number of fundamental meshes in network

2.3 Basic Concepts and Definition of Ventilation Network Theory

A mine ventilation system can be represented mathematically by a network. This network consists of the pathways of air flow and data associated with them. The following definitions are adopted from Wang (1983). The pathways, also called *branches*, are represented by lines and are interconnected among themselves at the points called *nodes*. A collection of nodes and branches is called a *linear graph*. The following topological properties of a linear graph will be used in the dissertation.

For each branch in the network, one of the two directions is assigned as a reference direction, which is indicated by an arrow. Associated with a branch there are two endpoints. The one to which the arrow is pointing is called the *final node* of a branch, and the other endpoint is called the *initial node*. A *path* is a sequence of branches in which all nodes are distinct and the final node of one branch is the initial node of the next branch. A *chain* is the nondirectional counterpart of a path and is applied to a network when the direction of its branches is disregarded. A closed chain is called a *mesh*.

A *spanning tree* or *tree* of a network is a connected subnetwork that contains all nodes of the network but no meshes. The branches in a tree are called the *tree branches*, and the remaining branches of the network are called the *chords*. Since the number of branches in a tree is always one less than that of the nodes, there are $n - 1$ tree branches and $m = b - n + 1$ chords.

Incidence Matrix

A branch is called an incidence with its initial and final nodes. The incidence matrix, denoted by A_a, of a network is a matrix of order n × b. If $A_a = [a_{ij}]$, the value of a_{ij} are defined as follows:

$a_{ij} = 1$ if branch j is incidence at node i and is directed away from node i

$a_{ij} = -1$ is branch j is incidence at node i and is directed toward node i

$a_{ij} = 0$ if branch j is not incidence at node i

<u>**For example**</u>, the incidence matrix of the ventilation network in Figure 1 is given by

$$A_a = \begin{array}{c|cccccccc|l} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & \leftarrow\text{branch} \\ \hline 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & \\ 2 & -1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & \\ 3 & 0 & -1 & -1 & 1 & 1 & 0 & 0 & 0 & \\ 4 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 1 & \\ 5 & 0 & 0 & 0 & 0 & -1 & -1 & 1 & 0 & \\ \uparrow & & & & & & & & & \\ \text{nodes} & & & & & & & & & \end{array} \qquad (2.1)$$

Observe that each column of A_a contains exactly two nonzero elements, one +1 and one –1. Since the column of A_a has this property, we may remove one row without losing any information. The (n – 1) × b matrix obtained by deleting a row from A_a is denoted by A and is called the reduced incidence matrix. All rows of A are linearly independent, and hence its rank is n – 1. The node corresponding to the selected row of A_a is referred to as the *reference node* of the network.

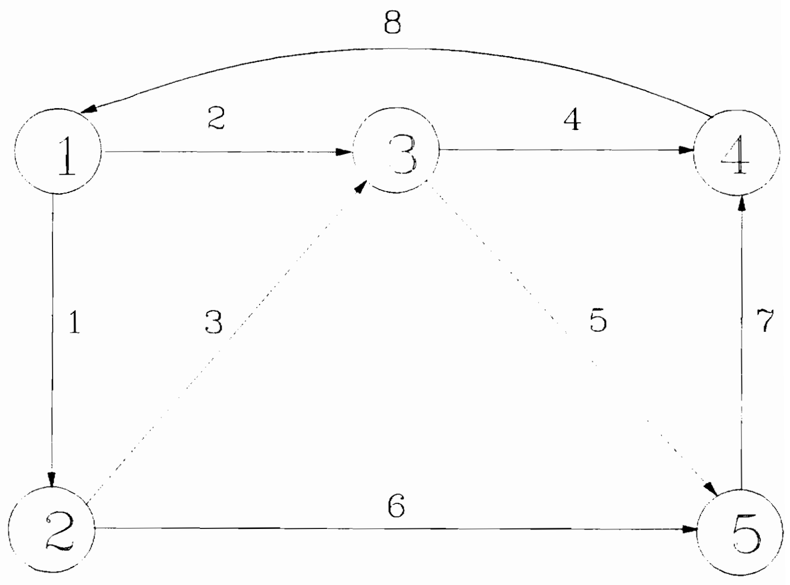


Figure 1. An Example Network

Fundamental Mesh Matrix

The fundamental meshes of a network with respect to a tree are the m meshes, each being formed by a chord and a unique chain in the tree connecting two endpoints of the chord. The direction of the fundamental meshes is chosen to agree with that of the defining chord. Each fundamental mesh contains only one chord, and each chord is contained in just one mesh. Mathematically, the fundamental meshes of a network with respect to a tree can be represented by the fundamental mesh matrix B. If $B = [b_{ij}]$, then the elements of b_{ij} are defined as follows:

$b_{ij} = 1$ if branch j is contained in mesh i and has the same direction

$b_{ij} = -1$ if branch j is contained in mesh i and has the opposite direction

$b_{ij} = 0$ if branch j is not contained in mesh i

<u>For example</u>, consider the network in Figure 1. If we choose a tree consisting of branches 1, 2, 5, 8 as shown in Figure 2, the fundamental meshes with respect to the tree can be expressed in terms of branch numbers as follows:

Mesh 1: 1, 3, –2

Mesh 2: 4, 8, 2

Mesh 3: 6, –5, –2, 1

Mesh 4: 7, 8, 2, 5

The corresponding fundamental mesh matrix is given by

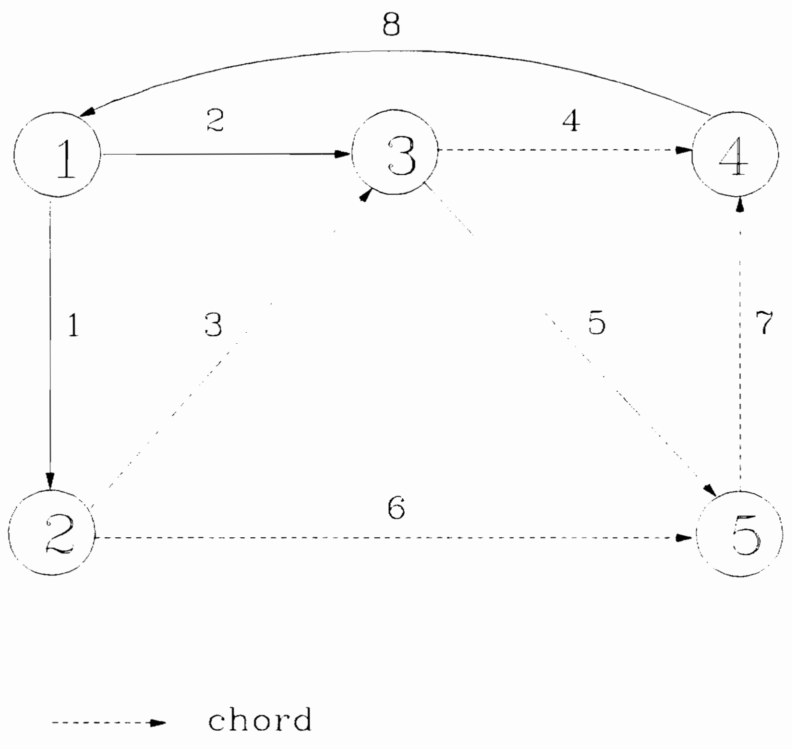


Figure 2. The Fundamental Meshes With Respect to a Tree

$$B = \begin{array}{c|cccccccc|l} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & \leftarrow \text{branch} \\ \hline 1 & 1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & \\ 2 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & \\ 3 & 1 & -1 & 0 & 0 & -1 & 1 & 0 & 0 & \\ 4 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & \\ \uparrow & & & & & & & & & \\ \text{mesh} & & & & & & & & & \end{array} \qquad (2.2)$$

Fundamental Cutset Matrix

A *cutset* of a network is a minimum collection of branches whose removal separates the network into two parts. Some examples of cutsets are illustrated in Figure 3. Each set of branches intersected by a broken line is a cutset. The direction of a cutset can be either from N_1 to N_2 or N_2 to N_1, where N_1 and N_2 are the two sets of nodes partitioned by the removal of the cutset from the network. The fundamental cutsets of the network with respect to a tree are the n_c cutsets, each of which contains only one tree branch, where $n_c = b - m = n - 1$. The direction of the fundamental cutset is chosen to coincide with that of the tree branch contained in the cutset. The fundamental cutset matrix, denoted by C, of a network with respect to a tree is an $n_c \times b$ matrix of rank n_c, such that if $C = [c_{ij}]$, then the elements of c_{ij} are defined as follows:

$c_{ij} = 1$ if branch j is contained in cutset i and has the same direction

$c_{ij} = -1$ if branch j is contained in cutset i and has the opposite direction

$c_{ij} = 0$ if branch j is not contained in cutset i

<u>For example</u>, if we choose the same tree used for the fundamental meshes in the

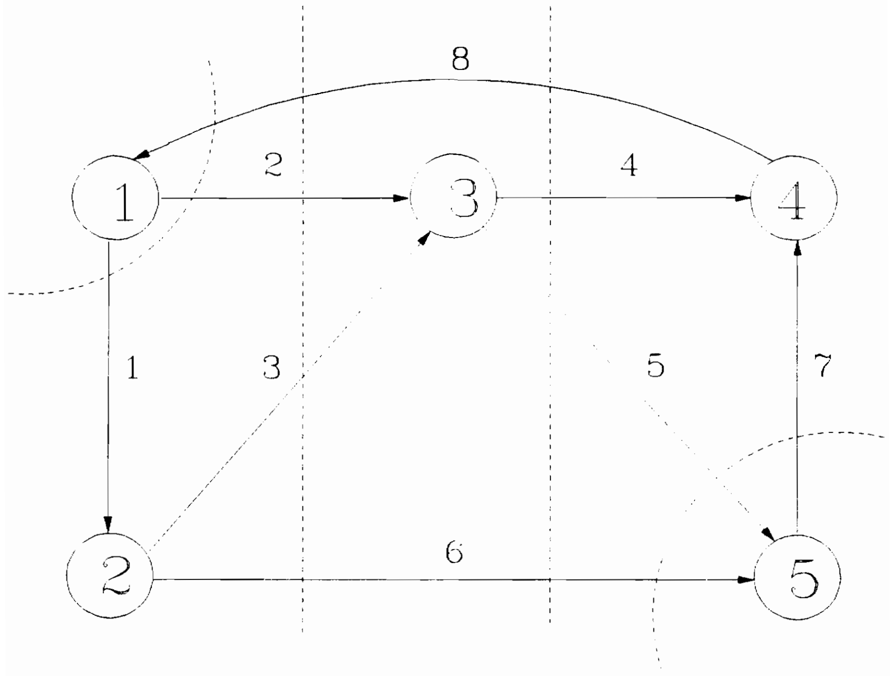


Figure 3. Some Examples of Cutsets

network in Figure 2, then the fundamental cutsets with respect to the tree are shown in Figure 4. The corresponding fundamental cutset matrix is given by

$$C = \begin{array}{c|cccccccc|l} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & \leftarrow \text{branch} \\ \hline 1 & 1 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & \\ 2 & 0 & 1 & 1 & -1 & 0 & 1 & -1 & 0 & \\ 3 & 0 & 0 & 0 & 0 & 1 & 1 & -1 & 0 & \\ 4 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 1 & \\ \uparrow & & & & & & & & & \\ \text{set} & & & & & & & & & \end{array} \qquad (2.3)$$

Observe in Figure 4 or Eq. 2.3 that each mesh "cut" by a cutset has two branches in common with the cutset.

Relationship among the Matrices of a Network

If the columns of the matrices A, B, and C of a network are arranged in the same branch order, as Eqs. (2.1) to (2.3), then the following relationships are valid:

$$AB^T = BA^T = 0 \qquad (2.4)$$

$$BC^T = CB^T = 0 \qquad (2.5)$$

where T indicates the transpose of a matrix.

By numbering the chords from 1 to m and the tree branches from m + 1 to b, and also designating the mesh that contains chord i by the subscript i (i = 1, 2,...,

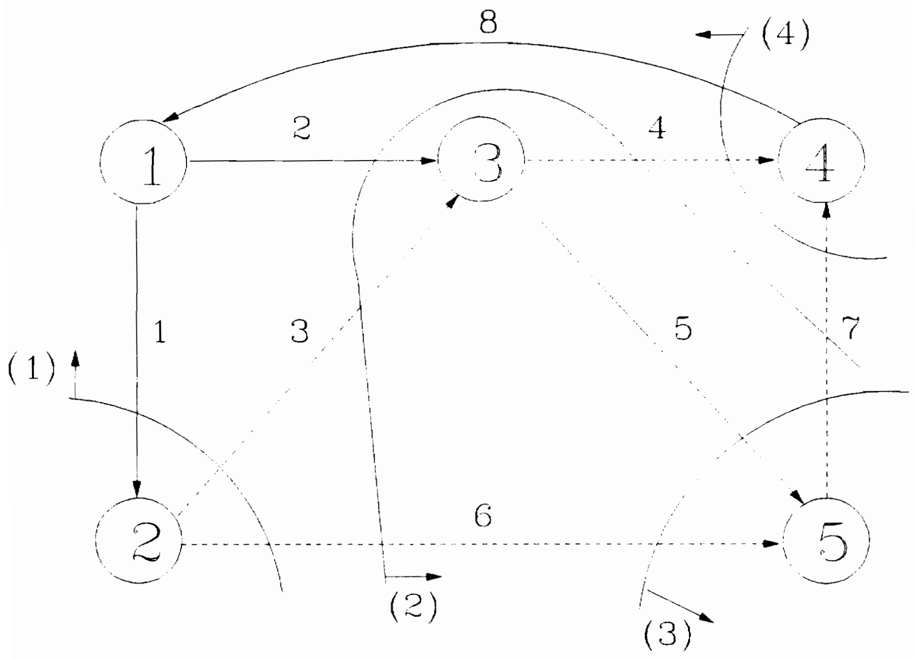


Figure 4. The Fundamental Cutsets With Respect to a Tree

m), the fundamental–mesh matrix takes the form

$$B = [I_m \;\; B_{12}] \tag{2.6}$$

where I_m is the identity matrix of order m. Similar to the matrix B, if the branches are numbered as in Eq. (2.6), and the cutset that contains branch m + i is numbered with i (i = 1, 2,..., n_c), then the fundamental cutset matrix can be expressed as

$$C = [C_{11} \;\; I_{n_c}] \tag{2.7}$$

where I_{n_c} is the identity matrix of order n_c. With the same arrangement of the branch order, the basis–incidence matrix A is partitioned as

$$A = [A_{11} \;\; A_{12}] \tag{2.8}$$

where A_{11} and A_{12} are submatrices consisting of columns for chords and tree–branches, respectively.

Rewriting the matrices B of Eq. (2.2) and C of Eq. (2.3) in the form of Eqs. (2.6) and (2.7), respectively, we have

$$B = \begin{array}{c} \\ \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{cccc|cccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ (3) & (4) & (6) & (7) & (1) & (2) & (5) & (8) \\ \hline 1 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & -1 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{array} \tag{2.9}$$

and

$$C = \begin{array}{c|cccc|cccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ & (1) & (2) & (5) & (8) & (3) & (4) & (6) & (7) \\ \hline 1 & 1 & 0 & 0 & 0 & -1 & -1 & 0 & 0 \\ 2 & 0 & 1 & 0 & 0 & 1 & 1 & -1 & -1 \\ 3 & 0 & 0 & 1 & 0 & 0 & 1 & -1 & -1 \\ 4 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & -1 \end{array} \qquad (2.10)$$

If the matrices A, B and C, respectively, are put in the form of Eqs. (2.6) to (2.8), the application of Eqs. (2.4) and (2.5) results in

$$B = [I_m \quad -A_{11}^T(A_{12}^{-1})^T] \qquad (2.11)$$

$$C = A_{12}^{-1}A = [-B_{12}^T \quad I_{n_c}] \qquad (2.12)$$

$$B_{12} = -A_{11}^T(A_{12}^{-1})^T \qquad (2.13)$$

$$C_{11} = -B_{12}^T \qquad (2.14)$$

2.4 Location Restrictions

When installing ventilation control devices, one has to consider health and safety regulations, availability of electrical power, and restrictions imposed by production requirements. Therefore, mine ventilation engineers must identify these restrictions on the location of ventilation control devices and accordingly specify the branches that can allow fan installation or regulator installation. According to these restrictions, the branches can be categorized into four groups. First, the branches allow installation of a fan or a regulator. Second, the branches allow installation of a fan but not a regulator. Third, the branches allow installation of a regulator but not a fan. Fourth, the branches do not allow the installation of a fan or a regulator. These restrictions must be input into the problem.

2.5 Natural Splitting Network Problem

The natural splitting network problem can be described as follows: for given resistance factors R_j for all branches and given fan locations and fan heads or fan characteristic curves, find air quantities Q_j for all branches, which satisfy the Kirchhoff's Laws:

$$\sum_{j=1}^{b} a_{ij} Q_j = 0 \qquad \text{for } i = 2, 3, \ldots, n$$

$$\sum_{j=1}^{b} b_{ij}(R_j |Q_j| Q_j - HF_j - HN_j) = 0 \qquad \text{for } i = 1, 2, \ldots, m$$

This is a system of b simultaneous equations with n − 1 linear and m nonlinear equations. It can be solved by the Hardy Cross method.

2.6 Controlled Splitting Network Problem

In this problem, the air quantities for all branches in the network are defined, or the air quantities for m branches corresponding to a set of chords are given and those for tree branches are uniquely determined by Eq. (2.19). Resistance factors R_j and natural ventilation pressure HN_j are all given. The unknowns to be solved are regulator head losses HR_j and fan heads HF_j. Mathematically, it can be presented in the following model:

$$\text{Minimize} \quad Z = \sum_{j=1}^{b} q_j HF_j = \sum_{j=1}^{b} q_j (HL_j + HR_j - HN_j)$$

subject to

$$\sum_{j=1}^{b} b_{ij}(HL_j + HR_j - HN_j - HF_j) = 0 \qquad i = 1, \dots, m$$

$$HR_j \geq 0 \qquad HF_j \geq 0 \qquad \text{for } j = 1, 2, \dots, b$$

where $HL_j = R_j |q_j| q_j$

Note that the objective function and the constraints are linear i.e., it is a linear programming problem.

2.7 Generalized Ventilation Network Problem

For the generalized ventilation network, air flow quantities for some branches are known. Resistance factors R_j and natural ventilation pressure HN_j are all given. Unknowns are air flow quantities for the remaining branches and fan heads HF_j and regulator head losses HR_j. It should be noticed that this problem is different from the natural splitting network and controlled splitting network problems because of its unfixed control device sizes and undetermined flow distribution.

2.7.1 *Objective Function*

An objective function is formulated to minimize the overall cost of ventilation, which includes operation cost, ownership cost and maintenance cost. It can be either in terms of an average annual cost of the system or in terms of the net present value of the cost of the system.

Assuming that annual power consumption, maintenance costs are directly related to fan power consumption, the objective function can be mathematically stated as

$$\text{Minimize } Z = \sum_{j=1}^{L} (C_p + C_m)\, Q_j HF_j + \sum_{j=1}^{L} C_j$$

where $C_p, C_m,$ = annual energy and ownership and maintenance costs, respectively.

C_j = annualized installation cost.

$L = \{j\text{: branch } j \text{ is allowed to have fan}\}$

2.7.2 Constraints

Power Restriction

From the operational and practical point of view, the pressure of fan installed in a branch of the network could not be below a given level with respect to the certain flow quantity. In other words, Q_jHF_j should be larger than the minimum permissible power. Mathematically, it says:

$$Q_jHF_j \geq P_j$$

or

$$HF_j \geq P_j/q_j$$

where P_j = minimum permissible horse power for branch j.

First Law

Like an electrical network, a ventilation network must satisfy Kirchhoff's current law, i.e., the mass flow out of any node is equal to the flow into that node. Mathematically, Kirchhoff's first law for mine ventilation networks can be expressed as

$$\sum_{j=1}^{b} a_{ij}Q_j = 0 \qquad \text{for } i = 2,\ldots, n \qquad (2.15)$$

or $$AQ = 0 \qquad (2.16)$$

where Q is a column matrix, given by

$$Q^T = [Q_1, Q_2, \ldots, Q_b] \tag{2.17}$$

and Eq. (2.15) indicates that node 1 was chosen as the reference node.

When a ventilation network satisfies Kirchhoff's current law, air quantities for all branches can be expressed as a function of chosen independent air quantities, and column matrix Q in Eq. (2.17) may be partitioned as

$$Q = \begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix} \tag{2.18}$$

where Q_1 is a column matrix of all air quantities corresponding to the chords of a tree, and Q_2 is a column matrix of all air quantities corresponding to the tree branches of the same tree. Substituting this expression into Eq. (2.16) for Q and using Eqs. (2.11) and (2.13), we obtain

$$Q_2 = B^T_{12} Q_1 \tag{2.19}$$

$$Q = B^T Q_1 \tag{2.20}$$

or, using the fundamental cutset matrix,

$$Q_2 = -C_{11} Q_1 \tag{2.21}$$

Notice that if air quantities for a set of chords with respect to a tree of a network are arbitrarily chosen, and Eq. (2.19) or (2.21) is used to obtain air quantities for all tree branches, then the network automatically satisfies Kirchhoff's current law.

<u>For example</u>, consider the network and the tree in Figure 1 and renumber the branches as shown in Figure 5, where branches 1, 2, 3, and 4 are chords. The fundamental mesh matrix is given by Eq. (2.9). If $q_1 = 2$, $q_2 = 4$, $q_3 = 1$, and $q_4 = 2$, then, from Eq. (2.19),

$$\begin{bmatrix} q_5 \\ q_6 \\ q_7 \\ q_8 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ -1 & 1 & -1 & 1 \\ 0 & 0 & -1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} = \begin{bmatrix} q_1 + q_3 \\ -q_1+q_2-q_3+q_4 \\ -q_3 + q_4 \\ q_2 + q_4 \end{bmatrix} = \begin{bmatrix} 3 \\ 3 \\ 1 \\ 6 \end{bmatrix} \qquad (2.22)$$

Note that the above matrix operation is equivalent to

$$q_j = \sum_{i=1}^{m} b_{ij} q_i \qquad \text{for } j = m + 1, m + 2, \ldots, b \qquad (2.23)$$

<u>Second Law</u>

Like an electrical network, a ventilation network must also satisfy Kirchhoff's voltage law i.e., the sum of the pressure drops around any mesh in the network must be equal to zero.

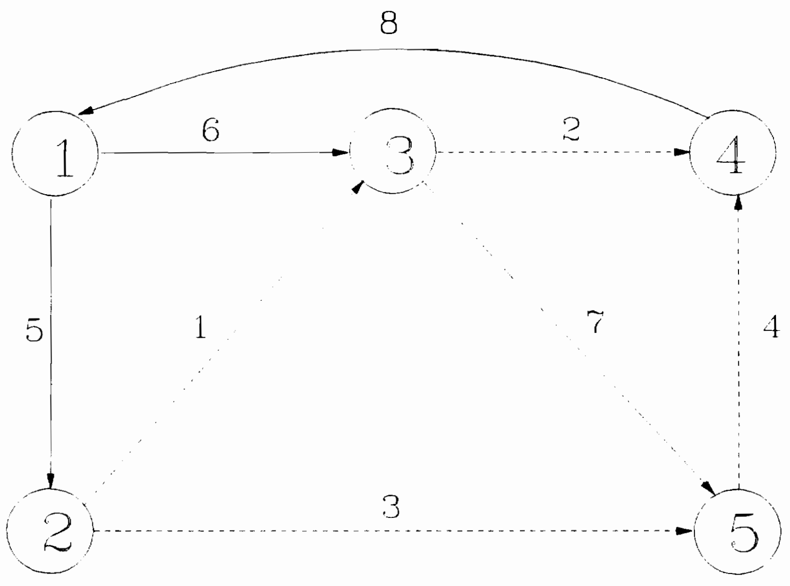


Figure 5. Renumbering of the Branches for the Network of Fig.1.

Mathematically, it can be expressed as

$$\sum_{j=1}^{b} b_{ij}H_j = 0 \qquad \text{for } i = 1,\ldots, m$$

or $BH = 0$

where $H_j = HL_j + HR_j - HF_j - HN_j$ and H is a column matrix, given by

$$H^T = [H_1, H_2,\ldots, H_n]$$

<u>Lower Bounds on Branch Flows</u>

The lower bounds on the branch flows are predetermined based on considerations of law, health and safety, and operating experience.

For all branches, the flow must be greater than or equal to a specified lower bound,

$$L_j \leq Q_j$$

Some branch flows may not have a lower bounds. Then L_j may be set equal to negative infinity (a very large negative number in computer code).

<u>Upper Bounds on Branch Flows</u>

The upper bounds on branch flows are predetermined in the same way as lower bounds. For all branches, the flow must be less than or equal to a specified upper bound,

$$Q_j \leq U_j$$

The upper bounds on the flows usually consider the flow velocity. For example, flow along an underground coal mine beltway is limited by certain velocity. If the cross section area is known, the velocity of the flow is

$$V = Q/\text{area}$$

Thus, given a maximum allowed velocity, the corresponding flow can be calculated as

$$U = Q_{max} = V_{max} \times \text{area}$$

<u>Nonnegative Constraints</u>

The decision variables HF_j and HR_j are nonnegative.

$$HF_j \geq 0 \qquad HR_j \geq 0 \qquad\qquad \text{for } j = 1,\ldots, b$$

2.7.3 Leakage

Leakage may occur at fan installations, in caved ground, in shafts between downcast and upcast compartments, at stoppings or doors in crosscuts between adjoining airways, and at stoppings and doors in main airways, particularly near the shaft at other surface openings.

Leakage acts as a split between intake and return airways of the ventilation circuit. The amount of leakage occurring is a function of the pressure. Therefore, leakage in the network can be treated by: allowing some leakage branches, giving them resistance factors, and treating them in the same way as other branches.

2.7.4 Problem Formulation

Summarizing this chapter, the following mathematical program formulation presents the problem of optimization of ventilation control device location and size in an underground mine ventilation network.

$$\text{Minimize } Z = \sum_{j=1}^{L} (C_p + C_m) Y_j Q_j HF_j + \sum_{j=1}^{L} C_j Y_j$$

subject to

$$Q_j HF_j \geq P_j \qquad j \in L$$

$$\sum_{j=1}^{b} a_{ij} Q_j = 0 \qquad i = 2, \ldots, n$$

$$\sum_{j=1}^{b} b_{ij}(R_j |Q_j| Q_j + HR_j - HF_j - HN_j) = 0 \qquad i = 1, \ldots, m$$

$$L_j \leq Q_j \leq U_j \qquad j = 1, \ldots, b$$

$HR_j \geq 0$ $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad$ $j=1,\ldots,b$

Y_j is 0 or 1, $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad$ for $j \in L$

The problem above is large, nonlinear, and nonconvex. The objective function contains some bilinear and binary terms.

CHAPTER III LITERATURE REVIEW

The optimization methods for ventilation network analysis and design are based on operations research algorithms and computers. Early work was directed toward the natural splitting network problem. Researchers used the Hardy Cross iteration method to determine air flow rates and pressure drops in the network where the fan pressure characteristic curve is replaced by a constant pressure line. This work mainly served to analyze an existing network or proposed network. For instance, it could be used to analyze whether or not a ventilation plan satisfied the requirement of air flow for each working district. According to the results, the designer could modify the plan to meet the flow requirement until a satisfactory air flow distribution was found. In the past three decades, this research has progressed greatly. It has made full use of the fast calculation capacity of computers and has incorporated thermodynamic algorithms to better simulate compressible flow, the effects of mine fires, and combination of airflows with climatic predictions.

Starting at beginning of the 1970s, some researchers applied operations research methods to the controlled splitting network problem. By formulating the problem as a linear programming model, linear programming, network programming, and graph theory can be used to solve this kind of problem.

Compared with natural and controlled splitting network problems, the semi–controlled splitting network problem is more complicated, because it can only be modeled through nonlinear programming. There is no universally applicable solution for it. Several methods have been used for the problem, namely the Hardy Cross iteration method, the flow path identification technique, the transformation of a semi–controlled network to a controlled network, the simulation, the nonlinear equation algorithm, and the nonlinear network optimization technique. These methods will be reviewed in this chapter.

The Hardy Cross Iteration Method (Wang, 1982)

The Hardy Cross iteration method can be used for the semi–controlled splitting network problem under the following assumptions:

(1) $1 \leq n_q \leq m$, and $n_q + n_f \leq m$

where n_q and n_f are, respectively, the numbers of fixed quantity branches and fan branches.

(2) the fixed quantity branches and fan branches can be contained in a set of chords with respect to a tree.

(3) main fan location and fan head or fan characteristic curve are given.

(4) booster fans and regulators are only located in the fixed quantity branches.

Since $n_q + n_f \leq m$, each of them is a chord and contained in only one fundamental mesh. The method numbers the meshes containing fixed quantity branches with 1 to n_q and fan branches with $n_q + 1$ to $n_q + n_f$. Then it performs the Hardy Cross iteration for $i = n_q + 1, ..., m$. For $i = 1,..., n_q$, calculate

$$t_j = \sum_{j=1}^{b} b_{ij}(R_j|q_j|q_j - HN_j)$$

If $t_j > 0$, $HR_j = t_j$ and $HF_j = 0$. Otherwise, $HR_j = 0$ and $HF_j = -t_j$.

Therefore, the solution only corresponds to the given main fan head and the above assumptions.

The Flow Path Identification Technique (Wang and Yao, 1984)

In 1984, Wang and Yao introduced a method which can reduce power consumption by choosing the regulator location in a ventilation network. The method is based on the following observation and concepts:

For an underground coal mine ventilation network, the airways can be grouped into an intake section, a face section and a return section. The face section separates the intake and return sections. There may also be some airways, associated with the face section, where the air quantities are related to the face section, when this is taken into account, the intake and return sections form closed networks.

The airflow distribution follows the two rules: (1) the intake air must flow through a face branch prior to flowing through the return; (2) face branches can not be connected in series.

For a network, there are b − n + 1 flow paths defined as follows: a flow path is a route through the intake, the face and the return where the direction of air flow is always in the same direction, which is in contrast to a path around a closed mesh. The pressure drop along each flow path between any common intake and return nodes should be identical.

If only regulators are considered, the conventional method of control would be to install regulators in the face branches with low pressure losses until the losses along all the flow paths were the same, i.e., equal to that of the flow path with the highest pressure loss. However, this does not change the air quantity distribution and only raises the pressure loss along the low pressure loss flow path without decreasing the pressure loss in the high pressure loss flow path. Regulating airways in the return or intake section can change the flow distribution and if this reduces the quantity in a branch of the high pressure loss flow path, the pressure loss will also be reduces. So will the power consumption.

Based on these observations, the criteria set up by Wang and Yao for selecting a regulator location in the return section are: (1) a regulator should not be in a maximum pressure loss flow path and (2) the airflow from the branch that is being regulated should flow directly into a maximum pressure loss flow path (conversely, the air should flow from a maximum pressure loss flow path into the branch being regulated if the regulator is located in an intake section). Based on these rules and criteria, a procedure was developed to calculate the resistance of regulator on the chosen locations. It is shown in Figure 6.

Simulation (Jones et al., 1986; Calizaya et al., 1987)

Simulation is essentially a trial and error method. For predetermined fan locations, the method searches for optimum combination of main fan and booster fan pressures. In order to do that, the simulator uses the Hardy Cross iterative method to simulate every possible combination between main fan and booster fan pressures, i.e., it considers every potential location of the fans as well as all possible values of the fan pressures. Theoretically, the alternatives could be infinite. To simplify the computation, the method searches for the relationship between fan pressure and regulator resistance. For example, for a network with three fixed flow branches, one main fan and one booster fan, the technique obtained the results shown in Figure 7. These results indicate that for each fixed quantity branch, the fan pressure–regulator resistance relationships are nearly linear and essentially have the same slope. Therefore, they assumed that they are contained in one plane.

The relationships can be represented by:

$$HF_m = \alpha HF_b + \beta HR_a + \gamma \qquad (3.1)$$

where

HF_m = main fan pressure

HF_b = booster fan pressure

HR_a = added resistance

α = rate of change of main fan with booster fan pressure

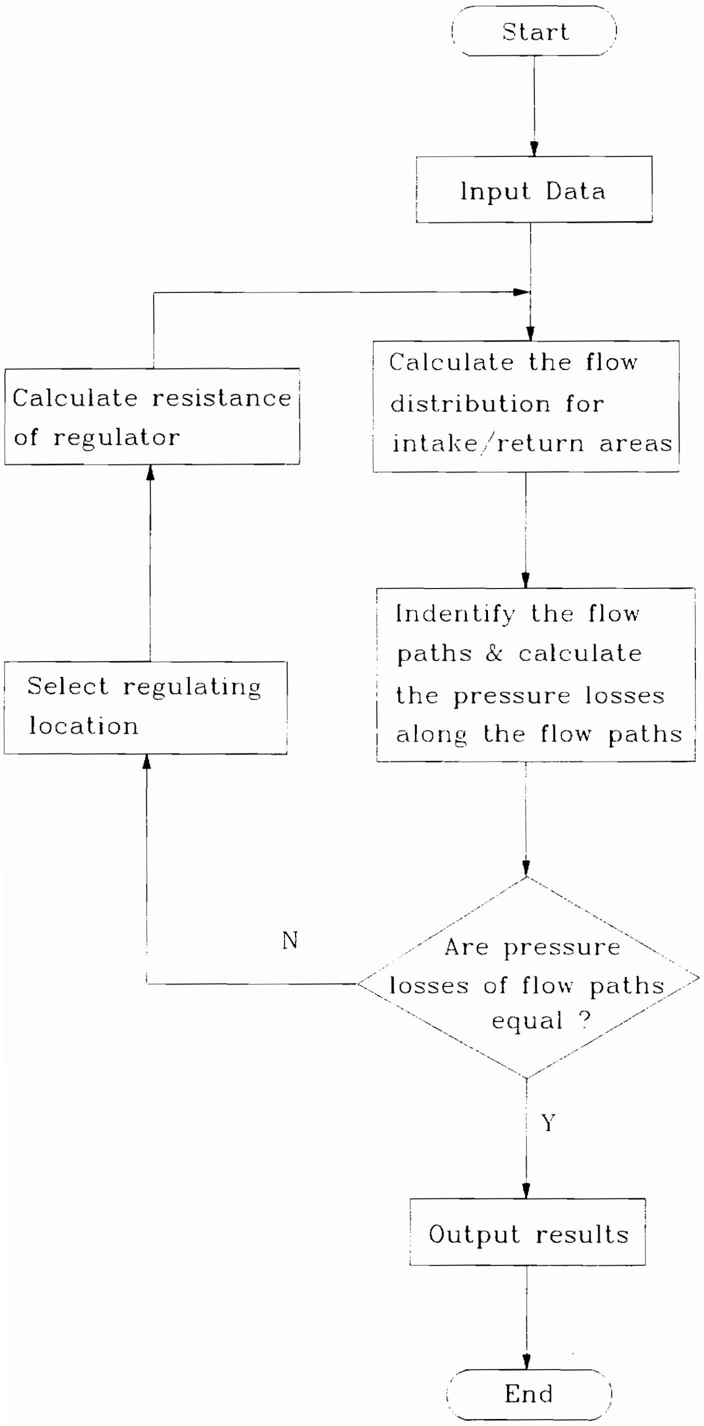

Start
Input Data
Calculate the flow distribution for intake/return areas
Indentify the flow paths & calculate the pressure losses along the flow paths
Are pressure losses of flow paths equal ?
N
Y
Select regulating location
Calculate resistance of regulator
Output results
End

Figure 6. The Flow Chart of Flow Path Identification Procedure

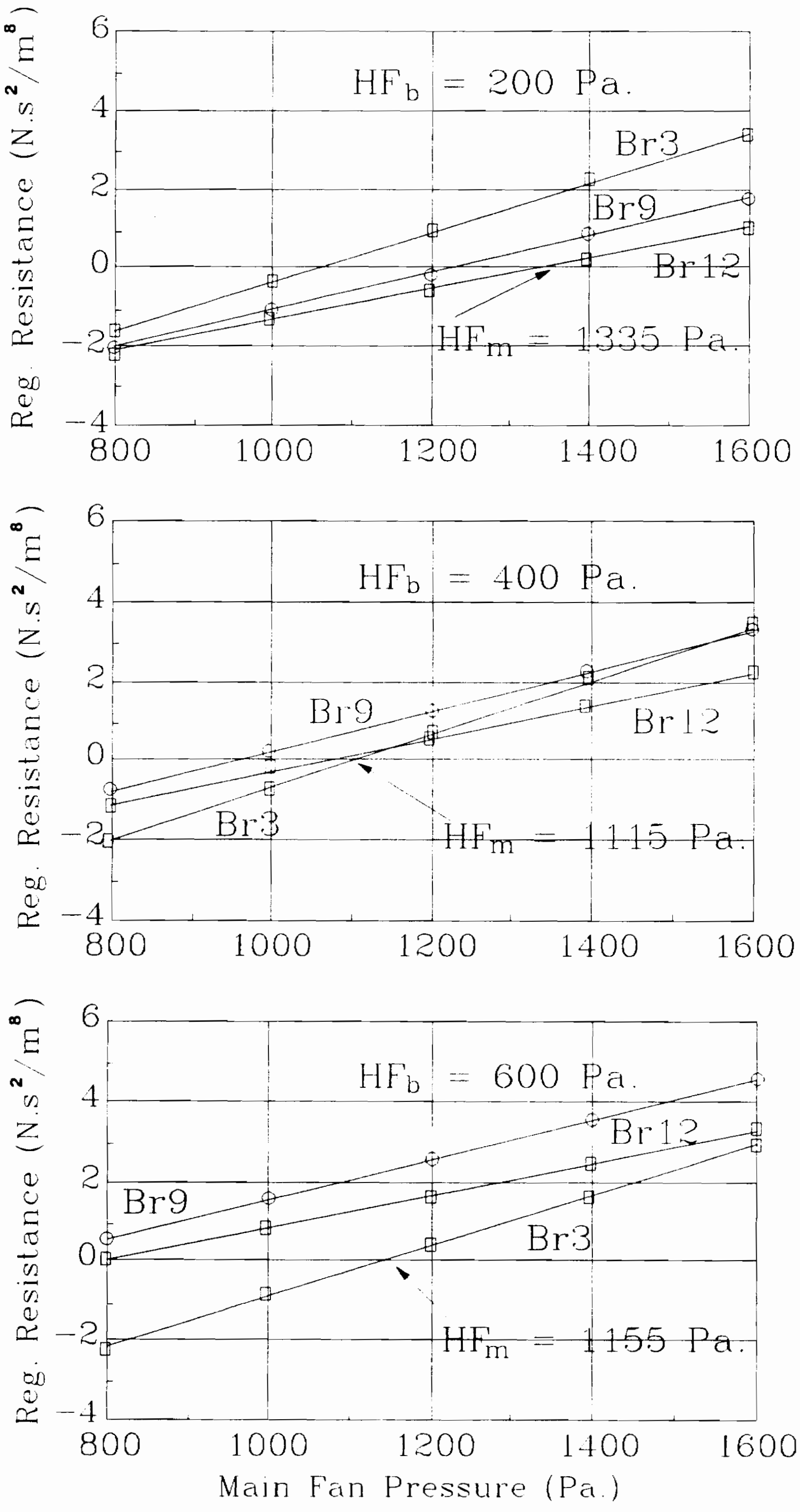


Figure 7. Regulator Resistance - Main Fan Pressure Relationship for Three Booster Fan Settings

β = rate of change of fan pressure with regulator resistance

γ = intercept

Three independent sets of HF_m, HF_b, and HR_a are sufficient to evaluate the parameters of equation (3.1).

For each fixed quantity branch, the method calculates the plane by using the Hardy Cross method for three different booster fan pressure values. Therefore, the intercept of these planes where $HR_a = 0$ is the optimum combination. For instance, the above network has a optimum combination as shown in Figure 8.

If there are several potential locations for the booster fan, the method must implement the procedure for each location and then compare the solutions.

If more booster fans are added, the relationship between fan pressure and regulator resistance will be more complicated. Moreover, regulators and fans may be located in other branches than the fixed quantity branches.

The Nonlinear Network Optimization Algorithm (Barnes, 1983)

Barnes' method minimized power consumption and formulated the ventilation network problem as follows:

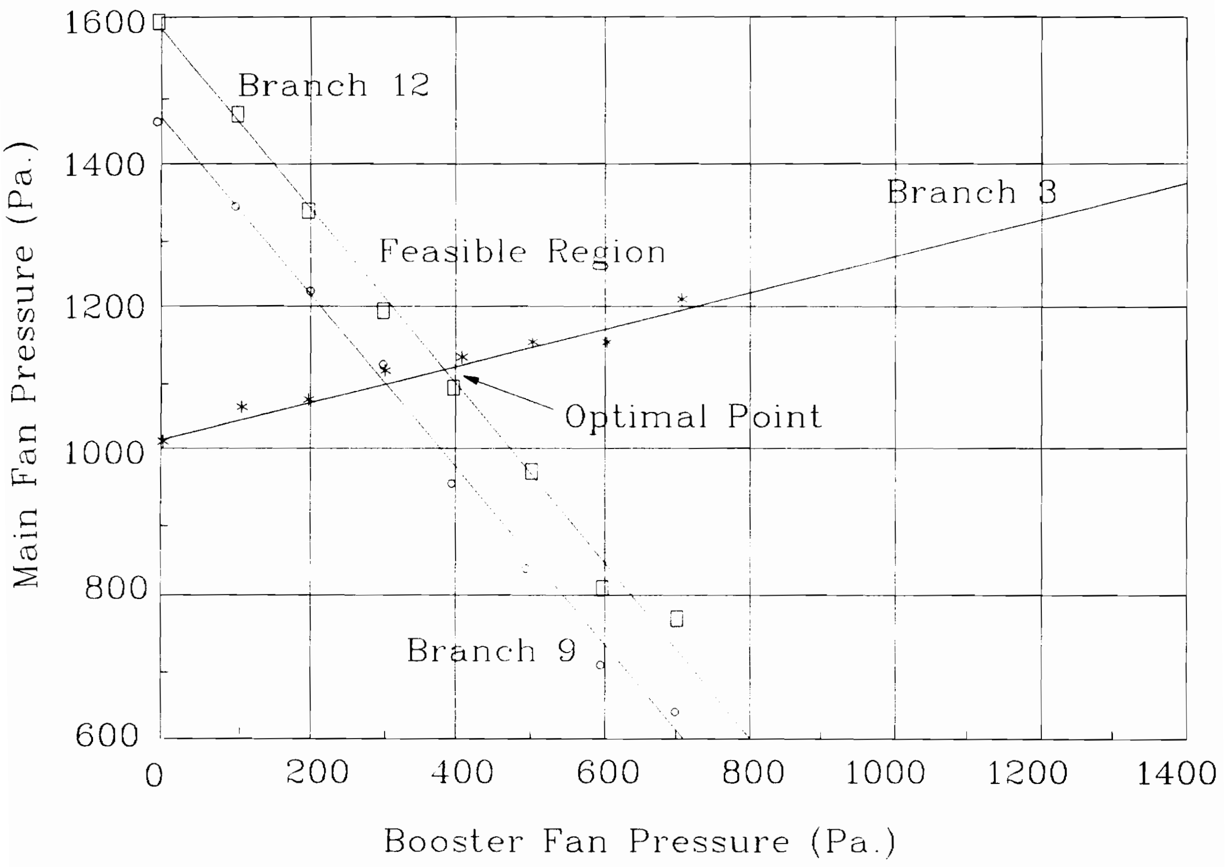


Figure 8. Optimal Combination of Fan Pressure For a Two Fan System

General Problem

$$\text{Minimize Power} = \sum_{i,j} Q_{ij} HF_{ij}$$

subject to

$$AQ = 0$$

$$A^tP + HF - HR - HL = 0$$

$$Q \geq L$$

$$Q \leq U$$

where subscript (i,j) stands for branch (i,j) with initial node i and final node j.

P = vector of nodal pressures

$HL_{ij} = R_{ij} | q_{ij} | q_{ij}$

Then the general problem was decomposed into two subproblems, namely Flow Problem and Pressure Problem as follows:

Flow Problem

Minimize power $= \phi(Q)$

subject to $AQ = 0$

$L \leq Q \leq U$

Pressure Problem

$$\text{Minimize} \quad \sum_{i,j} q_{ij} HF_{ij}$$

$$\text{subject to } A^t P + HF - HR = HL$$

Note that in the Pressure Problem, air flow quantity, q_{ij}, is known from the Flow Problem. So, it is the controlled flow network problem and can be solved by the out–of–kilter or network simplex methods in which its dual is solved. Similarly, the undefined function $\phi(Q)$ is given by its counterpart. It can be solved by Direction Finding and One Dimension Line Search techniques. The algorithm begins with an initial value for air flow distribution. Then it iterates back and forth between them until a predefined convergence criterion is met. The procedure is described as follows:

Step 0 Initialize the flow vector to a feasible flow, i.e., to solve the following problem

$$\text{Minimize} \quad \sum_{i,j} Q_{ij}^2$$

$$\text{subject to} \quad AQ = 0$$

$$L \leq Q \leq U$$

Let the initial solution be $Q^{[old]}$, set μ equal to twice the maximum flow in the any branch of the network.

Step 1 Fix the current flow vector and solve the resulting Pressure Problem.

Step 2 Calculate the gradient of the Pressure Problem objective function evaluated at the current feasible flow, P, HF, and HR.

the gradient $g_{ij} = 2R_{ij}|q_{ij}|\, x_{ij} + HF_{ij}$

where x and HF are the optimal values from the current dual of Pressure Problem computed in Step 1.

Step 3 Solve the following Direction Finding Problem for an improved feasible direction

$$\text{Minimize} \quad \sum_{i,j} g_{ij}Q_{ij}$$

$$\text{subject to} \quad AQ = 0$$

$$L \leq Q \leq U$$

Let the optimal solution to this problem be $Q^{[dir]}$.

Step 4 Carry out an optimal One Dimensional Line Search

$$\underset{0 \leq \alpha \leq 1}{\text{Minimize}}\ \phi(Q^{[new]})$$

where $Q^{[new]} = (1-\alpha)Q^{[old]} + \alpha Q^{[dir]}$

If the change in Q is greater than a tolerance, set $Q^{[old]} = Q^{[new]}$ and return to Step 1.

Step 5 If any artificial fans or regulators are significant then double μ and return to Step 1; otherwise stop, the current Q, HF, HR are optimal.

In this procedure, once the initial solution to one of the subproblems is determined, the rest of the computation uses a "hill climbing" technique. Because the problem is a nonconvex program, theoretically proper local solutions are possible. So, the procedure guarantees a convergence to a partial optimal solution rather than a global solution.

The Nonlinear Equation Algorithm (Wang, 1989)

The nonlinear equation algorithm's method is to solve a system of nonlinear equations by using the Newton method. This system consists of m nonlinear equations from Kirchhoff's second law

$$\sum_{j=1}^{b} b_{ij}(R_j|Q_j|Q_j + HR_j - HF_j - HN_j) = 0 \qquad i = 1,\ldots,m$$

and n – 1 linear equations from Kirchhoff's first law

$$\sum_{j=1}^{b} a_{ij}Q_j = 0 \qquad i = 1,\ldots, n-1$$

So, the total number of equations is the number of branches (m = b – n + 1).

If a ventilation control device (a fan or a regulator) is conceptually associated with every branch in the network, each branch has two variables, the air flow quantity and control device pressure loss. Therefore, the system of nonlinear equations has 2 b variables. If there are b variables independent in the system, the other b variables can be expressed in terms of the b known variables. In other words, we can solve the system of nonlinear equations with b independent variables and b dependent variables by techniques for dealing with a system of nonlinear equations such as the Newton Raphson method.

It should be noticed that the system can be solved under the condition that the number of independent variables is b.

This study also explored the condition under which the solution is not unique based on graph theory.

The Transformation of a Semi–controlled Network to a Controlled Network (Wu and Topuz, 1987)

Since it is easier to solve a controlled network problem than to solve a semi–controlled network problem, a method was developed to convert a semi–controlled network to a controlled network. The technique uses the same assumptions about the network structure as that in the Wang and Yao method. If the leakage is ignored, the total air flow quantity entering and leaving these closed networks is known. The distribution of air flow in the closed networks may vary within the lower and upper boundaries imposed upon the branches in the network. An ideal distribution of air flows, while satisfying the quantity limits on each branch, should minimize the power consumption:

$$\text{Minimize} \quad \sum_{i}^{nc} Q_i H_i = \sum_{i}^{nc} R_i |Q_i|^3$$

$$\text{subject to} \quad L_i \leq Q_i \leq U_i$$

where nc = the number of branches in the closed network

R_i = resistance for branch i

H_i = head loss for branch i

As seen above, the objective function is nonlinear and the network is a nonlinear network. It can be solved by the implicit piecewise approximation method in the convex minimum cost flow problem (Jensen et al., 1980). The details can be seen in Wu and Topuz.

After distribution of air flow in the intake and return section is determined, the flow quantity in each branch is known. As a result, the semi–controlled network is converted to a controlled network. Then it can be solved by the algorithms dealing with the controlled network problem. It should be noticed that the assumptions about the network structure limit the utilization of this algorithm.

CHAPTER IV FOUNDATION FOR THE SOLUTION

4.1. Introduction

The solution algorithm presented in this dissertation for optimizing ventilation control device location and size in an underground mine ventilation system is an optimization technique for the nonconvex programming problem. It can find global optimal solution to general classes of nonconvex optimization problem. This algorithm formulates the problem in terms of the special ordered sets of variables i.e., it linearizes the nonlinear functions occurring in either the objective function or constraints of the problem by the special ordered sets of variables. For the linearized problem, the algorithm incorporates the automatic interpolation technique into the branch and bound procedure to reach the optimal solution. In other words, the algorithm solves the problem by the Simplex Method, then interpolates other special ordered sets of variables when necessary. After that, it checks for optimality, if found, stop; otherwise branch on the special ordered sets of variables directly to partition the problem into subproblems. For the subproblems, it repeats the same procedure as above until the optimal solution is reached.

4.2. Problem Statement

The general separable nonlinear nonconvex programming problem can be expressed as follows:

$$\text{Minimize } Z = \sum_{j\in L} f_j(x_j) + \sum_{j\notin L} f_j(x_j)$$

subject to

$$\sum_{j\in L} g_{ij}(x_j) + \sum_{j\notin L} g_{ij}(x_j) \{<, =, >\}\ b_i \qquad i = 1,\dots, m \qquad (4.1)$$

$$l_j \le x_j \le u_j \qquad \text{for } j \notin L$$

$$x_j \ge 0 \qquad \text{for } j \in L$$

where x_j is decision variable, $f_j(x_j)$ is the function of x_j and $g_{ij}(x_j)$ is the ith row function of x_j, $L = \{j\text{: } f_j \text{ and } g_{ij} \text{ for } i = 1,\dots, m \text{ are linear}\}$ i.e., L is a subset where f_j and g_{ij} are linear function. The objective function and the constraint functions can be expressed as the sum of functions, each involving only one variable, which is referred to as separable, and they could be nonconvex, and x_j is either a variable of the problem or else some linear function of other variables of the problem. If Z is replaced by $-x_0$, the problem changes into (Beale, 1981)

$$\text{Maximize} \quad x_0$$

subject to

$$x_0 + \sum_{j\in L} f_j(x_j) + \sum_{j\notin L} f_j(x_j) = 0 \qquad (4.2)$$

$$\sum_{j\in L} g_{ij}(x_j) + \sum_{j\notin L} g_{ij}(x_j) \{<, =, >\}\ b_i \qquad \text{for } i = 1,\dots, m$$

$$l_j \le x_j \le u_j \qquad \text{for } j \notin L$$

$x_j \geq 0$ for $j \in L$

or Maximize x_0

subject to

$$x_0 + \sum_{j \in L} g_{0j}(x_j) + \sum_{j \notin L} g_{0j}(x_j) = 0 \qquad (4.3)$$

$$\sum_{j \in L} g_{ij}(x_j) + \sum_{j \notin L} g_{ij}(x_j) \{<, =, >\} b_i \qquad \text{for } i = 1,\ldots, m$$

$$l_j \leq x_j \leq u_j \qquad \text{for } j \notin L$$

$$x_j \geq 0 \qquad \text{for } j \in L$$

where $g_{0j}(x_j) = f_j(x_j)$

4.3. Special Ordered Sets

As seen in Eq. (4.3), some nonlinear function $g_{ij}(x_j)$ for $i = 0, 1,\ldots, m$ may be involved in the ith constraint. If $g_{ij}(x_j)$ is approximately piecewise linear between the points $x_j = x_{j0}, x_{j1}, \ldots, x_{jk}$, then we can proceed as follows (Beale et al., 1976).

Define a set of nonnegative variables λ_{jk} and the constraints

$$\sum_k \lambda_{jk} = 1 \qquad \text{for } j \in L \qquad (4.4)$$

$$\sum_k x_{jk}\lambda_{jk} - x_j = 0 \qquad \text{for } j \in L \qquad (4.5)$$

and then the nonlinear function $g_{ij}(x_j)$ is represented by the linear function

$$\bar{g}_{ij}(x_j) = \sum_k g_{ij}(x_{jk})\lambda_{jk} \quad i = 0, 1, \ldots, m,\ j \notin L \qquad (4.6)$$

provided that, for each j, not more than two of λ_{jk} are allowed to be nonzero and if there are as many as two they are adjacent. The variables λ_{jk} are referred to as *special ordered sets*. Eq.(4.4) is called *convexity row.* Eq.(4.5) is called *reference row*, because this row is referenced to which to refer when calculating the effective argument x_j from the weights λ_{jk}.

This concept can be illustrated geometrically. In Fig.9, g(x) is a continuous curve. It is approximated by a set of connected line segments. The set of six line segments that approximate g(x) by $\bar{g}(x)$ is designated. It can readily be seen that the approximation $\bar{g}(x)$ is such that $\bar{g}(x) = g(x)$ at the end points of each of line segment. Consider the line segment between x_1 and x_2. In the interval $[x_1, x_2]$, one can represent $\bar{g}(x)$ by

$$\bar{g}(x) = g(x_1) + \frac{g(x_2) - g(x_1)}{x_2 - x_1}(x - x_1) \qquad x_1 \leq x \leq x_2 \qquad (4.7)$$

Similarly, we can represent the line segment $\bar{g}(x)$ in interval $[x_2, x_3]$ by

$$\bar{g}(x) = g(x_2) + \frac{g(x_3) - g(x_2)}{x_3 - x_2}(x - x_2) \qquad x_2 \leq x \leq x_3 \qquad (4.8)$$

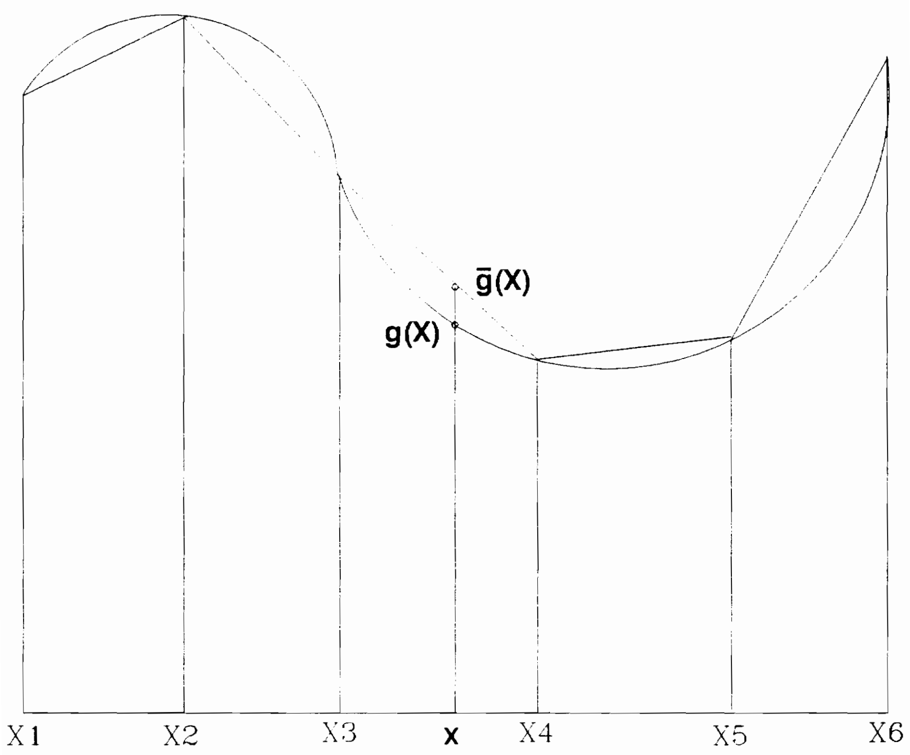


Figure 9. Piecewise Linear Approximation of a Nonlinear Function

Similar expressions can be written for the line segments in each of the other intervals.

Eq.(4.7) demonstrates that in the interval $[x_1, x_2]$ the fraction $(x - x_1)/(x_2 - x_1)$ is a number between 0 and 1 for any value of x between x_1 and x_2. Therefore, for any value of x in the interval we can designate this fraction λ and write

$$\begin{aligned}\bar{g}(x) &= g(x_1) + [g(x_2) - g(x_1)]\lambda \\ &= g(x_1) - \lambda g(x_1) + \lambda g(x_2) \\ &= (1 - \lambda)g(x_1) + \lambda g(x_2)\end{aligned}$$

If we designate $\lambda_2 = \lambda$ and $\lambda_1 = 1 - \lambda$, we have

$$\bar{g}(x) = \lambda_1 g(x_1) + \lambda_2 g(x_2) \qquad x_1 \leq x \leq x_2$$

$$\lambda_1 + \lambda_2 = 1 \tag{4.9}$$

$$\lambda_1 \geq 0, \quad \lambda_2 \geq 0$$

From $(x - x_1)/(x_2 - x_1) = \lambda = \lambda_2$, we have

$$x = \lambda(x_2 - x_1) + x_1$$

$$= (1 - \lambda)x_1 + \lambda x_2 \tag{4.10}$$

$$= \lambda_1 x_1 + \lambda_2 x_2$$

In a similar way, considering the intervals $[x_2, x_3]$ and $[x_3, x_4]$, it is possible to write

$$\bar{g}(x) = \lambda_2 g(x_2) + \lambda_3 g(x_3) \qquad x_2 \leq x \leq x_3$$

$$\bar{g}(x) = \lambda_3 g(x_3) + \lambda_4 g(x_4) \qquad x_3 \leq x \leq x_4$$

where $\lambda_2 + \lambda_3 = 1$ and $\lambda_3 + \lambda_4 = 1$, depending upon which interval x is in. Combining these representations, one can represent $\bar{g}(x)$ over $[x_1, x_4]$ by

$$\bar{g}(x) = \lambda_1 g(x_1) + \lambda_2 g(x_2) + \lambda_3 g(x_3) + \lambda_4 g(x_4)$$

$$\lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 = 1 \qquad (4.11)$$

$$\lambda_1, \lambda_2, \lambda_3, \lambda_4 \geq 0$$

and where $x = \lambda_1 x_1 + \lambda_2 x_2 + \lambda_3 x_3 + \lambda_4 x_4$

If, in addition to the above, the following two conditions are imposed: (1) at most two λ_k can be positive, (2) only adjacent λ_k are allowed to be positive, then the result is that Eqs. (4.11) will represent $\bar{g}(x)$, no matter which interval, $[x_1, x_2]$, $[x_2, x_3]$, $[x_3, x_4]$, x is in. Therefore, Eqs. (4.11) and the above two conditions allow us to represent g(x) by $\bar{g}(x)$ over the three intervals. The extension to any number of intervals is obvious. The illustration above shows that a function of a single variable g(x) can be represented by a set of connected line segments. In other words, a nonlinear function can be represented by an approximating linear function with special ordered sets of variables. As a result, it could be solved by a linear programming algorithm such as the Simplex Method. Using this concept and letting $g_{0j}(x_j) = f_j(x_j)$, the function $g_{ij}(x)$ for $i = 0, 1, \ldots, m$ and $j \notin L$ in Section 4.2 could be replaced by its linear approximation given below

$$\bar{g}_{ij}(x) = \sum_k \lambda_{jk} g_{ijk}(x_{jk}) \qquad \text{for } i = 0, 1, \ldots, m; \text{ and } j \notin L$$

$$\sum_k \lambda_{jk} = 1 \qquad \text{for } j \notin L$$

$$\lambda_{jk} \geq 0 \qquad k = 0, \ldots, K_j; \text{ and } j \notin L$$

$$\text{where } x_j = \sum_k x_{jk} \lambda_{jk} \qquad \text{for } j \notin L$$

The following problem can then be viewed as the problem that approximates the original problem in Section 4.2.

Maximize x_0

subject to

$$x_0 + \sum_{j \in L} g_{0j}(x_j) + \sum_{j \notin L} \bar{g}_{0j}(x_j) = 0 \qquad (4.12)$$

$$\sum_{j \in L} g_{ij}(x_j) + \sum_{j \notin L} \bar{g}_{ij}(x_j) \{ <, =, > \} b_i \qquad i = 1, \ldots, m$$

$$l_j \leq x_j \leq u_j \qquad \text{for } j \notin L$$

$$x_j \geq 0 \qquad \text{for } j \in L$$

Note that $\bar{g}_{ij}(x_j) = g_{ij}(x_j)$ for $i = 0, 1, \ldots, m$ and $j \in L$ and the functions $g_{ij}(x_j)$ for $i = 0, 1, \ldots, m$ and $j \notin L$ are piecewise linear. However, by using the definition of $\bar{g}_{ij}(x_j)$, the problem can be restated in an equivalent, more manageable form as follows:

Maximize $\quad x_0$

subject to

$$x_0 + \sum_{j \in L} g_{0j}(x_j) + \sum_{j \notin L} \sum_k \lambda_{jk} g_{0jk}(x_{jk}) = 0 \qquad (4.13)$$

$$\sum_{j \in L} g_{ij}(x_j) + \sum_{j \notin L} \sum_k \lambda_{jk} g_{ijk}(x_{jk}) \{ <, =, > \} b_i \qquad i = 1, \ldots, m$$

$$\sum_k \lambda_{jk} = 1 \qquad \text{for } j \notin L$$

$$\lambda_{jk} \geq 0 \qquad \text{for } k = 0, 1, \ldots, K_j; \text{ and } j \notin L$$

$$x_j \geq 0 \qquad \text{for } j \in L$$

Here it should be noted that the variable values to be determined are the λ_{jk} for $k = 0, 1, \ldots, K_j$ and $j \notin L$, and x_j for $j \in L$. The values of $g_{ijk}(x_{jk})$ for $j \notin L$ are the coefficients obtained by evaluating $g_{ij}(x_j)$ at a set of fixed points x_{jk}. The approximating problem is clearly a linear programming problem.

4.4 Transformation of the Non–separable Model

As seen in Section 4.2, the formulation of a nonlinear function demands that the nonlinear function be separable, i.e., they can be expressed as the sum of functions, each only involving one variable. But in some problems, variables are interdependent in such a way that product terms are found in the objective function and/or in the constraints. In some cases it is possible to convert the product term into a separable form so as to apply the special ordered sets for linearization.

For example: A quadratic form xy can be written as

$$xy = (x + y)^2/4 - (x - y)^2/4$$

By the simple transformation $z_1 = \dfrac{x + y}{2}$, $z_2 = \dfrac{x - y}{2}$,

we get $xy = z_1^2 - z_2^2$

which is a separable function in the variables z_1 and z_2. Then we can introduce nonnegative variables λ_k and μ_k to linearize them and write

$$\sum_k \lambda_k = 1$$

$$\sum_k \mu_k = 1$$

$$\sum_k \lambda_k z_{1k} - z_1 = 0$$

$$\sum_k \mu_k z_{2k} - z_2 = 0$$

So, product term xy can be represented by the following linear expression

$$\sum_k \lambda_k z_{1k}^2 + \sum_k \mu_k z_{2k}^2$$

For function yf(x), if we suppose that x must take one of the values x_k for $k = 0,\ldots,K$, and that $y_{min} \leq y \leq y_{max}$. We can introduce two sets of nonnegative variables λ_{1k} and λ_{2k} for $k = 0,\ldots,K$, and write (Beale et al., 1978)

$$\sum_k \lambda_{1k} + \sum_k \lambda_{2k} = 1$$

$$\sum_k \lambda_{1k} x_k + \sum_k \lambda_{2k} x_k - x = 0$$

$$\sum_k y_{min} \lambda_{1k} + \sum_k y_{max} \lambda_{2k} - y = 0$$

Then if $x = x_{k_1}$ we can represent the nonlinear function yf(x) by the linear function

$$\sum_k y_{min} f(x_k)\lambda_{1k} + \sum_k y_{max} f(x_k)\lambda_{2k}$$

if we impose the further restriction that $\lambda_{1k} = 0$ and $\lambda_{2k} = 0$ whenever $k \neq k_1$.

An alternative way of dealing with product terms in a nonlinear model is to use logarithms.

For example, for the problem of maximizing wxy, let

$$z = wxy$$

then $\log z = \log w + \log x + \log y$

Hence, the problem of maximizing wxy is equivalent to

Maximize z

subject to $\log z = \log w + \log x + \log y$

Assuming that w, x and y are all positive variables, then the problem is separable. More generally, any polynomial term, or, for that matter, any product of individually separable functions can be separated by means of logarithmic transformation as long as the arguments of all logarithms involved are restricted to positive values.

These transformations show that an expression can be made separable by introducing additional variables and additional constraints. They can be cumbersome, increasing the size of the model considerably as well as the computational difficulty.

4.5. Improved Branch and Bound Method

4.5.1 Basic Branch-and-Bound Concepts

The basic procedure of the branch–and–bound technique is the following (Hillier and Lieberman, 1986). Suppose that the objective function is to be maximized. Assume that a *lower bound* on the objective function is available. This lower bound normally is the value of the objective function for the best feasible solution identified thus far, which is referred to as the *incumbent solution.* The first step is to partition the set of all feasible solutions into several subsets, and then, for each one, an *upper bound* is obtained for the value of the objective function of the solutions within that subset. Those subsets whose upper bound is lower than the current lower bound on the objective function value are then excluded from further consideration. Other subsets are also discarded if they are found to be of no further interest, either because the subset has no feasible solution or because its best feasible solution has been found (so that this solution can be recorded and the rest of the subset eliminated). A subset that is excluded from further consideration for any of these reasons is said to be *fathomed.* After the appropriate subsets have been fathomed, each of the remaining subset is partitioned further into several subsets. Their upper bounds are obtained in turn and used as before to exclude some of these subsets from further consideration. From all the remaining subsets, another one is selected for further partitioning, and so on. This process is repeated again and again until a feasible solution whose objective function value is no less than the upper bound for any remaining subset is found. Such a feasible solution must be optimal because none of the subsets can contain a better solution.

Summary of Branch and Bound Technique

Initialization step Set lower bound $Z_l = -\infty$. Begin with the entire set of solutions under consideration (including any infeasible solutions that cannot conveniently be eliminated) as the only remaining subset. Before beginning the regular iterations through the following steps, apply just the bound step, and the optimality test step to this one subset.

Branch step Use some branch rule to select one of the remaining subsets (those neither fathomed nor partitioned) and partition it into two or more new subsets of solutions.

Bound step For each new subset, obtain a upper bound Z_u on the value of the objective function for the feasible solution in the subset.

Fathom step For each new subset, exclude it from further consideration if

Fathoming test 1. $Z_u \leq Z_l$

or

Fathoming test 2. The subset is found to contain no feasible solutions, or

Fathoming test 3. The best feasible solution in the subset has been identified (so Z_u corresponds to its objective function value); if this situation occurs and $Z_u > Z_l$, then $Z_l = Z_u$, store this solution as the new incumbent solution and reapply Fathoming Test 1 to all remaining subsets.

Optimality test step When there are no remaining subsets; the current incumbent solution is optimal. Otherwise, return to the branch step.

4.5.2 Notation and Terminology

The subscript i refers to a row, and the subscript j refers to a special ordered set. The columns within a special ordered set are defined as a function of the reference row entry x_j, rather than identifying them by subscripts which can not easily be given meaningful numerical values. So the variables in the jth set are denoted by $x_j(x_j)$. This corresponds to the variable denoted by λ_{jk} if $x_j = x_{jk}$. In any linear programming subproblem, more than one member of the set may take nonzero values, so it is still convenient to use the subscript k to distinguish between different values of x_j. The optimal values of the $x_j(x_{jk})$ in any linear programming subproblem are denoted by $x_j^0(x_{jk})$.

The vector of coefficients of the variable $x_j(x_{jk})$ is denoted by $g_j(x_{jk})$, and its ith component, representing the coefficient in the ith row, is denoted by $g_{ij}(x_{jk})$.

Let i_{rj} denote the reference row for the jth set. Since the coefficient of $x_j(x_j)$ in this row is x_j, this results in the equation

$$g_{i_{rj}j}(x_j) = x_j$$

The symbol Σ_k denotes summation over all variables in the jth set. In this notation the convexity row normally takes the form

$$\sum_k x_j(x_{jk}) = 1$$

However, some formulations require the more general form

$$\sum_k x_j(x_{jk}) + S_{sj} = b_{sj}$$

where S_{sj} is a nonnegative slack variable. Therefore, this equation can be used for the more general type of convexity row, although in practice usually $b_{sj} = 1$ and $S_{sj} = 0$.

$\bar{x}_j$ is defined as *average reference row entry* for the jth special ordered set. This can be interpreted as the value of the jth argument implied by the solution to the linear programming subproblem. A formula that applies in all cases is

$$\bar{x}_j = \sum_k x_j^0(x_{jk})x_{jk} / \sum_k x_j^0(x_{jk}) = \sum_k x_j^0(x_{jk})x_{jk}/(b_{sj} - S_{sj}^0)$$

Corresponding to any linear programming subproblem, three approaches to the contribution of the variables in the jth set to the problem can be defined. There is the vector g_{aj} of *actual contributions* defined by

$$g_{aj} = \sum_k x_j^0(x_{jk})g_j(x_{jk})$$

There is the vector g_{cj} of *corrected contributions* corresponding to the average reference row entry defined by

$$g_{cj} = (\sum_k x_j^0(x_{jk}))g_j(\overline{x}_j) = (b_{sj} - S_{sj}^0)g_j(\overline{x}_j)$$

A third approach is relevant in deciding where to branch in the jth set. If x_{js} and x_{je} denote the smallest and largest values of x_j for which $x_j^0(x_j) > 0$, then we define the vector $g_{I_j}(x_j)$ of *interpolated coefficients* corresponding to any x_j between x_{js} and x_{je} by

$$g_{I_j}(x_j) = (1-\theta)g_j(x_{js}) + \theta g_j(x_{je})$$

where θ is defined by the equation

$$x_j = (1-\theta)x_{js} + \theta x_{je}$$

The *discrepancy* between $g_{I_j}(x_j)$ and $g_j(x_j)$ can be regarded as an indication of the extent to which the current linear programming subproblem misrepresents the consequences of giving the jth argument the value x_j.

As usual π_i denotes the *shadow price* on the ith row. This refers to the ultimate objective function if the linear programming subproblem is feasible, and to the sum of infeasibilities otherwise. We denote the *reduced cost* of the variables $x_j(x_j)$ by $d_j(x_j)$ (see APPENDIX A for detail), so

$$d_j(x_j) = \sum_i \pi_i g_{ij}(x_j)$$

The term *degradation* is used to mean the reduction in the value of the objective function in a subproblem caused by imposing the additional condition that a special ordered set must be satisfied i.e., that the members of the special ordered sets take feasible values.

4.5.3 The Improved Branch and Bound Method

If f(x) is a convex function, any combination of non–neighboring λ variables will overestimate it. Therefore there need not be concern about invalid combinations occurring in the optimal solution to the problem if either f(x) is part of an objective function or if f(x) occurs on the left–hand–side of a less than–or–equal–to inequality (see APPENDIX B for theorem and proof). When such a convex problem occurs, the function f(x) can be approximated by a piecewise linear function and then the simplex method can be used to solve the problem.

If convexity conditions do not apply, a global optimum solution can be found by the branch and bound technique. The problem is first solved as an ordinary linear programming problem, without imposing any of the restrictions about neighboring λ

variables. If the resulting solution satisfies the additional conditions on the λ variables, then all is well. Otherwise some index, such as r, is chosen. A general procedure for finding a suitable pair of variables λ_r, λ_{r+1} for branching is to compute

$$\overline{w} = \sum_k w_k \lambda_k / \sum_k \lambda_k$$

where the weights w_k are the reference row entries x_k if they exist and the sequence numbers k otherwise. The ends of the current interval, and hence the pair $(\lambda_r, \lambda_{r+1})$ are determined by finding w_r and w_{r+1} such that

$$w_r \leq \overline{w} \leq w_{r+1} \qquad \text{or} \qquad w_r \geq \overline{w} \geq w_{r+1}$$

Then the original problem is replaced by two subproblems. In one subproblem, the additional requirement

$$\lambda_0 = \lambda_1 = \dots = \lambda_{r-1} = 0$$

is imposed and in the other subproblem the additional requirement is imposed that

$$\lambda_{r+1} = \lambda_{r+2} = \dots = \lambda_k = 0$$

Clearly, any valid solution of the original problem must be a feasible solution of one or other of these subproblems. Therefore the subproblems may now be solved, and if the optimum solution to either subproblem still fails to satisfy the conditions, it can be subdivided further in the same way. Thus a branch and bound tree is built which may be searched until we find the optimum solution to the problem. So, the final solution will not have more than two members of the set of λ variables with non–zero values, and if there are two such members they must be adjacent.

It may be noted that the accuracy of the procedure largely depends on the number of the special ordered set variables. However, as the number of the set variables increases, the size of the approximation linear problem will be enlarged. So, instead of having $K + 1$ variables λ_{jk} associated with $K + 1$ specific values of x_{jk} we consider a continuous infinity of variables λ_j associated with all possible values of x_j between x_{j0} and x_{jk}. The linear programming problem is solved by defining one of these variables explicitly only when it is required to "enter the basis" in the simplex method. The process is called *Automatic Interpolation*. It will be discussed in detail in next section.

The following questions remain:

(a) Which subproblem should be explored next?

(b) Which special ordered sets should be branched on?

(c) Once a particular special ordered set is selected for branching, where should the branch be made?

To select the next subproblem, after dividing the current problem into a pair of subproblems, choose the one with the higher upper bound on the objective function of the linear programming and store the other one on the end of the list of the alternative subproblems. When the problem is solved, take the last problem from the list as the current problem.

To answer part (b), calculate the degradation. A crude estimate of the degradation resulting from using the corrected contribution instead of the actual contribution is (see APPENDIX C for detail)

$$\sum_i \pi_i(g_{cij} - g_{aij})$$

This quantity may underestimate the degradation, since the shadow prices only measure the costs of infinitesimal changes in contributions to the corresponding rows, and may therefore underestimate the costs of finite changes in these contributions. In particular, the degradation estimated from this formula could be zero in situations where further branching is necessary.

To overcome this difficulty, two further sets of nonnegative input quantities P_i^+ and P_i^- should be defined, representing *pseudo shadow prices*. They should be such that $P_i^+ = 0$ whenever the ith row has a negative slack or is an objective function, P_i^- $= 0$ whenever the ith row has a positive slack or is an objective function. Otherwise P_i^+ and P_i^- should be strictly positive. Good estimates of these quantities should be used when available, but underestimates are less disastrous than overestimates so the default value should be set at 10^{-4}.

The approximate degradation D_{A_j} that will be incurred by removing the infeasibility in the jth set is then defined by

$$D_{A_j} = \sum_i \max\{ P_i^+(g_{cij} - g_{aij}), - P_i^-(g_{cij} - g_{aij}), \pi_i(g_{cij} - g_{aij})\}$$

or $D_{A_j} = \sum_i C_i$

where $C_i = \max(P_i^+, \pi_i)\,(g_{cij} - g_{aij})$ if $g_{cij} - g_{aij} \geq 0$

$C_i = \max(P_i^-, - \pi_i)\,|g_{cij} - g_{aij}|$ if $g_{cij} - g_{aij} < 0$

To decide which special ordered set to branch on, we follow the rule of branching on the one with the greatest estimated degradation.

To answer part (c), compare the coefficients $g_{ij}(x_j)$ with the interpolated coefficients $g_{I_{ij}}(x_j)$ and define the function $D_{I_j}(x_j)$ by

$$D_{I_j}(x_j) = \sum_i \max\{P_i^+(g_{ij}(x_j) - g_{I_{ij}}(x_j)), - P_i^-(g_{ij}(x_j) - g_{I_{ij}}(x_j)),$$

$$\pi_i(g_{ij}(x_j) - g_{I_{ij}}(x_j))\}$$

When branching on the jth set, it is natural to branch at a value x_j^m of x_j that maximizes $D_{I_j}(x_j)$. This maximizes the benefit from the division into two subproblems if it is measured by the average reduction in $D_{I_j}(x_j)$ over the current range of possible values of x_j. If this function has more than one local maximum, any for which $D_{I_j}(x_j) > 0$ can be accepted.

4.6. Automatic Interpolation (also see APPENDIX A)

As mentioned in last section, automatic interpolation is used to improve the accuracy to which each nonlinear function is approximated. In order to decide which variable to add, the most negative reduced cost (within some small tolerance) of all possible new vectors λ_{jk}, corresponding to values of x_j within its current bounds, must be found. This means finding a global minimum of the function

$$f_j = d_j(x_j) = \sum_i \pi_i g_{ij}(x_j) \qquad (4.14)$$

for the bounded scalar argument x_j. The method used to solve the problem will be discussed in APPENDIX D.

One variable from each set is introduced in this way, provided that the most negative reduced cost is sufficiently negative, whenever the subproblem has been optimized using the currently available variables. This process continues until either

(a) the bound falls below the value of the objective function at the best trial solution so far, in which case the subproblem can be abandoned,

(b) f is greater than or equal to the threshold value for all variables in all sets.

4.7. General Solution Algorithm

All of the preceding concepts may now be incorporated into a single computational algorithm. The requirements include a stored list of alternative problems to be solved, together with upper bounds on the objective function values for these problems, and a record of the best feasible solution currently known. A general procedure of the algorithm is then as follows. The flow chart for the algorithm is presented in Figure 10.

Initialization Step Linearize the problem in terms of special ordered set variables, let the incumbent solution $Z = -\infty$ and the linear problem be the current problem. Go to the Bound Step.

Bound Step Solve the current problem as a linear programming problem and then go to the Feasibility Test Step.

Feasibility Test Step If the solution is feasible, go to the Interpolation Step. If it is not feasible, fathom it and go to the Backtrack Step.

Interpolation Step Determine whether the current problem needs to be interpolated. If so, add one variable with the most negative reduced cost to the basis for each set, go to the Bound Step; otherwise, go to the Optimality Test Step.

Optimality Test Step If the solution is optimal (each of special ordered sets at most has 2 adjacent nonzero variables), fathom the problem, and if the solution is better than Z, let Z = the solution and eliminate all the alternatives with the objective function values less than or equal to Z on the list. Go to the Backtrack Step. If the solution is not optimal, go to the branch step.

Branch Step Compute the degradations and select the set with the greatest degradation to branch on, and then determine where to branch. Decompose the current problem into two subproblems. Let one be the current problem and put other one on the alternative subproblem list, go to the Bound Step.

Backtrack Step If the list is empty, stop: the current solution is optimal. Otherwise take the last problem from the list as the current problem and go to the Bound Step.

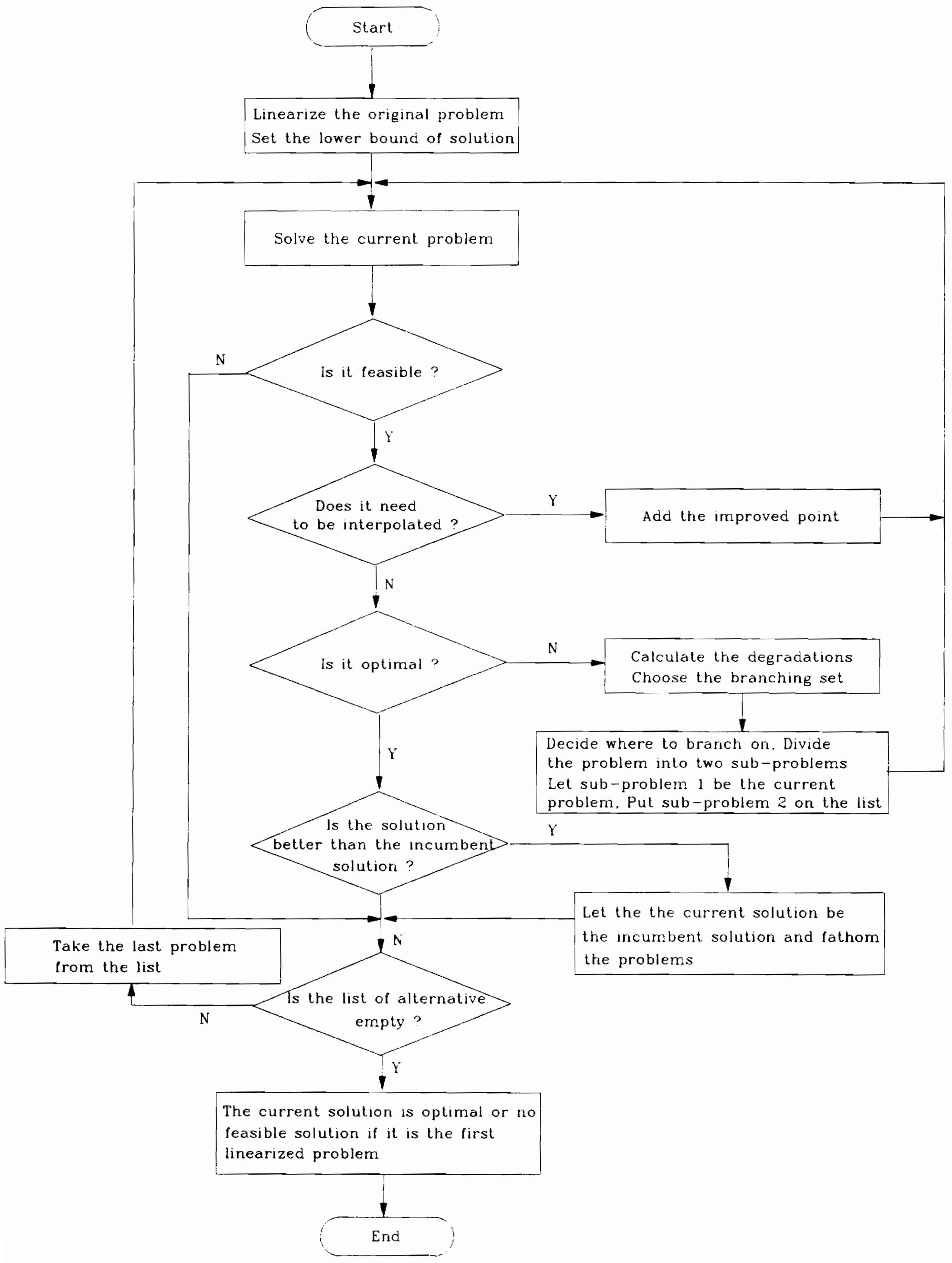

Start
Linearize the original problem
Set the lower bound of solution
Solve the current problem
Is it feasible ?
N
Y
Does it need to be interpolated ?
Y
Add the improved point
N
Is it optimal ?
N
Calculate the degradations
Choose the branching set
Decide where to branch on. Divide the problem into two sub-problems Let sub-problem 1 be the current problem. Put sub-problem 2 on the list
Y
Is the solution better than the incumbent solution ?
Y
Let the the current solution be the incumbent solution and fathom the problems
N
Take the last problem from the list
Is the list of alternative empty ?
N
Y
The current solution is optimal or no feasible solution if it is the first linearized problem
End

Figure 10. Flow Chart of the General Solution Algorithm

CHAPTER V PROBLEM SOLUTION

5.1 Problem Formulation

As seen in Section 2.7.4, the problem of optimizing ventilation control device locations and sizes in underground mine ventilation systems can be represented by the following program formulation:

$$\text{Minimize } Z = \sum_{j=1}^{L} (C_p + C_m) Y_j Q_j HF_j + \sum_{j=1}^{L} C_j Y_j$$

subject to

$$Q_j HF_j \begin{cases} = 0 & \text{if } Y_j = 0 \\ \geq P_j & \text{if } Y_j = 1 \end{cases}$$

$$\sum_{j=1}^{b} a_{ij} Q_j = 0 \qquad i = 2,\ldots,n \qquad (5.1)$$

$$\sum_{j=1}^{b} b_{ij}(R_j|Q_j|Q_j + HR_j - HN_j - HF_j) = 0 \qquad i=1,\ldots,m$$

$$L_j \leq Q_j \leq U_j$$

$$HR_j \geq 0$$

Y_j is 0 or 1, for $j = 1,\ldots,L$

For the model above, the possible subsets of solutions of Y_j are first defined and the partial solution is obtained for each subset. The whole set of the partial solutions consists of the combination of the elements in the set L. For example, assume that in a ventilation network, fans are allowed to be put in three branches, namely branches 1, 2, 3 and the main fan should be located in branch 1. There are four possible solutions for Y_j: $\{Y_1\}$, $\{Y_1, Y_2\}$, $\{Y_1, Y_3\}$, and $\{Y_1, Y_2, Y_3\}$. After we define the partial solution for Y_j, the problem becomes:

$$\text{Minimize } Z = \sum_{j=1}^{L_1} (C_p + C_m) Q_j HF_j + \sum_{j=1}^{L_1} C_j$$

subject to

$$Q_j HF_j \geq P_j \qquad \text{for } j \in L_1$$

$$\sum_{j=1}^{b} a_{ij} Q_j = 0 \qquad i = 2,\ldots,n \qquad (5.2)$$

$$\sum_{j=1}^{b} b_{ij}(R_j |Q_j| Q_j + HR_j - HN_j - HF_j) = 0 \qquad i = 1,\ldots,m$$

$$L_j \leq Q_j \leq U_j$$

$$HR_j \geq 0$$

where $L_1 = \{j: Y_j = 1\}$.

It could be noticed that C_p, C_m, and C_j are constants. Therefore, for each possible solution, the problem can be simplified as:

$$\text{Minimize } Z = \sum_{j=1}^{L_1} Q_j HF_j$$

subject to

$$Q_j HF_j \geq P_j \qquad \text{for } j \in L_1$$

$$\sum_{j=1}^{b} a_{ij} Q_j = 0 \qquad i = 2,\ldots,n \tag{5.3}$$

$$\sum_{j=1}^{b} b_{ij}(R_j | Q_j | Q_j + HR_j - HN_j - HF_j) = 0 \qquad i = 1,\ldots,m$$

$$L_j \leq Q_j \leq U_j$$

$$HR_j \geq 0$$

or

$$\text{Minimize } Z = \sum_{j \in L_2} Q_j HF_j + \sum_{j \in L_3} q_j HF_j$$

subject to

$$Q_j HF_j \geq P_j \qquad \text{for } j \in L_2$$

$$HF_j \geq P_j/q_j \qquad \text{for } j \in L_3$$

$$\sum_{j \in L_3} a_{ij} q_j + \sum_{j \notin L_3} a_{ij} Q_j = 0 \qquad i = 2,\dots,n \qquad (5.4)$$

$$\sum_{j \in L_3} b_{ij}(R_j q_j^2 + HR_j - HN_j - HF_j) = 0$$
$$+ \sum_{j \notin L_3} b_{ij}(R_j |Q_j| Q_j + HR_j - HN_j - HF_j) = 0 \qquad i = 1,\dots,m$$

$$L_j \leq Q_j \leq U_j$$

$$HR_j \geq 0$$

where $L_2 = \{j$: flow in branch j splits naturally and $j \in L_1\}$

$L_3 = \{j$:flow in branch j is fixed and $j \in L_1\}$

Thus problem (5.3) or (5.4) is solved corresponding to each partial solution of $\{Y_j\}$. Then the values of objective functions are compared. The solution with the minimum cost will be the optimal solution.

For the bilinear terms, QHF, suppose that the fan head pressure HF takes the value between its lower and upper bounds and two sets of variables λ_{1k}, λ_{2k} are introduced and defined

$$\sum_k \lambda_{1k} + \sum_k \lambda_{2k} = 1$$

$$\sum_k Q_k\lambda_{1k} + \sum_k Q_k\lambda_{2k} - Q = 0$$

$$\sum_k HF_{min}\lambda_{1k} + \sum_k HF_{max}\lambda_{2k} - HF = 0$$

Then we can approximately represent the function QHF by the linear function

$$\sum_k HF_{min}Q_k\lambda_{1k} + \sum_k HF_{max}Q_k\lambda_{2k}$$

For the nonlinear terms $R|Q|Q$ in the constraints, we use the special ordered set technique to linearize them and represent it as

$$\sum_k R|Q_k|Q_k\lambda_{1k} + \sum_k R|Q_k|Q_k\lambda_{2k} \qquad \text{if Q stands for flow rate in fan branch.}$$

or

$$\sum_k R|Q_k|Q_k\lambda_k \qquad \text{if Q is the flow rate in non fan branch.}$$

After these transformation above and let $L_4 = \{$j:flow splits naturally in the non fan branch j$\}$, the nonlinear nonconvex problem becomes the linear problem:

$$\text{Minimize} \sum_{j\in L_2} (\sum_k HF_{jmin}Q_{jk}\lambda_{j1k} + \sum_k HF_{jmax}Q_{jk}\lambda_{j2k}) + \sum_{j\in L_3} q_jHF_j$$

subject to

$$\sum_{j\in L_2} a_{ij} (\sum_k Q_{jk}\lambda_{j1k} + \sum_k Q_{jk}\lambda_{j2k}) + \sum_{j\in L_4} a_{ij}\sum_k Q_{jk}\lambda_{jk}$$

$$+ \sum_{j \in b-L_2-L_4} a_{ij} q_j \qquad \text{for } i = 2, \ldots, n$$

$$\sum_{j \in L_2} b_{ij}[R_j (\sum_k Q_{jk}|Q_{jk}|\lambda_{j1k} + \sum_k Q_{jk}|Q_{jk}|\lambda_{j2k}) + HR_j - HN_j - HF_j]$$
$$+ \sum_{j \in L_4} b_{ij}[R_j \sum_k Q_{jk}|Q_{jk}|\lambda_{jk} + HR_j - HN_j - HF_j]$$
$$+ \sum_{j \in b-L_2-L_3} b_{ij}(R_j q_{jk}^2 + HR_j - HN_j - HF_j) = 0 \qquad \text{for } i = 1, \ldots, m$$

$$\sum_k \lambda_{j1k} + \sum_k \lambda_{j2k} = 1 \qquad \text{for } j \in L_2$$

$$\sum_k \lambda_{jk} = 1 \qquad \text{for } j \in L_4$$

$$\sum_k HF_{jmin}\lambda_{j1k} + \sum_k HF_{jmax}\lambda_{j2k} - HF_j = 0 \qquad \text{for } j \in L_2$$

$$\lambda_{j1k}, \lambda_{j2k} \geq 0 \qquad \text{for } k = 0, 1, \ldots, K_j \text{ and } j \in L_2$$

$$\lambda_{jk} \geq 0 \qquad \text{for } k = 0, 1, \ldots, K_j \text{ and } j \in L_4$$

$$HR_j \geq 0 \qquad \text{for all } j$$

Here, the original variables Q_{jk} and HF_{jmin}, HF_{jmax} are constants and the new variables are λ_{j1k}, λ_{j12}, and λ_{jk}. Obviously, this is a linear programming model.

5.2 Algorithm Implementation

5.2.1 Model Analysis

Power Constraints

These constraints require that the horse power for each fan branch be larger than certain permissible values, which prevent locating some fans with unpractical fan head pressure in the allowable branches. Mathematically, it is expressed as:

$$Q_j HF_j \geq P_j \qquad j \in N_1$$

$$HF_j \geq P_j/q_j \qquad j \in N_2$$

If $L_j \leq Q_j \leq U_j$ and $HF_{jmin} \leq HF_j \leq HF_{jmax}$, then

$$Q_j HF_j \geq L_j HF_{jmin} \qquad j \in N_1$$

$$HF_j \geq HF_{jmin} \qquad j \in N_2$$

Flow Rates

Eq.(2.15) $\sum_{j=1}^{b} a_{ij} Q_j = 0 \qquad i = 2,\ldots,n$

or

(2.16) $AQ = 0$

is known as Kirchhoff's first law, where the air quantities are not all independent. Based on the graph theory in Section 2.7.2, dependent air quantities can be expressed as a function of independent air quantities i.e.,

$$Q_2 = B^T_{12} Q_1 \tag{2.19}$$

$$Q = B^T Q_1 \tag{2.20}$$

$$Q_2 = -C_{11} Q_1 \tag{2.21}$$

where Q_1 is a column matrix of all air quantities corresponding to the chords of a tree. For a semi–controlled network, only part of the elements in the matrix Q_1 are known (otherwise, it will be a controlled splitting network). As a result, only the elements of Q_2 that can be expressed by the independent elements in Q_1 are known, other are unknowns.

<u>**For example**</u>, consider Figure 5, where branches 1, 2, 3, and 4 are chords. If $Q_1 = q_1 = 2$, $Q_2 = q_2 = 4$, $Q_3 = q_3 = 1$, and Q_4 is unknown, from Eq. (2.19), we have

$$\begin{bmatrix} Q_5 \\ Q_6 \\ Q_7 \\ Q_8 \end{bmatrix} = \begin{bmatrix} Q_1 + Q_3 \\ -Q_1 + Q_2 - Q_3 + Q_4 \\ -Q_3 + Q_4 \\ Q_2 + Q_4 \end{bmatrix} = \begin{bmatrix} 3 \\ 1 + Q_4 \\ -1 + Q_4 \\ 4 + Q_4 \end{bmatrix}$$

where Q_5 is known because it can be expressed by known q_1 and q_3 while Q_6, Q_7, nd Q_8 are unknown since Q_4 is unknown. However, they can be expressed by Q_4. This expression may reduce the nonlinear terms. As a result, the special ordered sets can be reduced. For example, originally three special ordered sets for variables Q_5, Q_6, Q_7, and Q_8 need to be considered. After the implementation above, only one set for Q_4 is needed.

5.2.2 The problem Initialization

The algorithm in Chapter IV starts with only two vectors in each special ordered set at the end points of the permitted range and then uses automatic interpolation to create others as required.

<u>**For example**</u>, if we want to linearize nonlinear variable Q_j where $L_j \leq Q_j \leq U_j$, the starting special ordered sets variables are λ_{j0} and λ_{jk} corresponding to $Q_j = L_j$ and $Q_j = U_j$, respectively. The resulting convexity row, and reference row are:

$$\lambda_{j0} + \lambda_{jk} = 1$$

$$Q_j = \lambda_{j0} L_j + \lambda_{jk} U_j$$

The nonlinear function $g_j(Q_j) = R_j |Q_j| Q_j$ is approximately represented by

$$g_j(Q_j) = \lambda_{j0} R_j |L_j| L_j + \lambda_{jk} R_j |U_j| U_j$$

The bilinear function $Q_j HF_j$ is approximately represented by

$$HF_{jmin}L_j\lambda_{j10} + HF_{jmin}U_j\lambda_{j1k} + HF_{jmax}L_j\lambda_{j20} + HF_{jmax}U_j\lambda_{j2k}$$

where

$$\lambda_{j10} + \lambda_{j1k} + \lambda_{j20} + \lambda_{j2k} = 1$$

$$HF_{jmin}\lambda_{j10} + HF_{jmin}\lambda_{j1k} + HF_{jmax}\lambda_{j20} + HF_{jmax}\lambda_{j2k} - HF_j = 0$$

$$L_j\lambda_{j10} + U_j\lambda_{j1k} + L_j\lambda_{j20} + U_j\lambda_{j2k} - Q_j = 0$$

For the branch j which does not allow the installation of a fan, we set $HF_j = 0$. Similarly, for the branch that does not allow the installation of a regulator, we set $HR_j = 0$.

5.3 Special Cases

The methodology developed in this research can also be used to solve the natural splitting network problem and the controlled splitting network problem.

5.3.1 Natural Splitting Network Problem

For the natural splitting network problem, minimizing the total cost of ventilation is equivalent to minimizing power consumption and power constraints disappear because the fan heads are given. So, the problem becomes:

$$\text{Minimize} \quad Z = \sum_{j \in L} Q_j Hf_j$$

subject to

$$\sum_{j=1}^{b} a_{ij}Q_j = 0 \qquad i = 2,\dots, n$$

$$\sum_{j=1}^{b} b_{ij}(R_j|Q_j|Q_j - HN_j - Hf_j) = 0 \qquad i = 1,\dots, m$$

$$L_j \leq Q_j \leq U_j \qquad j = 1,\dots, b$$

where Hf_j indicates that the fan heads are known. This is a nonlinear nonconvex problem with the linear objective function and $n - 1$ linear constraints and m nonlinear constraints. Using special ordered sets variables λ_{jk} for $j = 1,\dots, b$ and following constraints,

$$\sum_{k} \lambda_{jk} = 1 \qquad j = 1,\dots, b$$

$$Q_j = \sum_{k} \lambda_{jk}Q_{jk} \qquad j = 1,\dots, b$$

$$\lambda_{jk} \geq 0 \qquad k = 0,\dots, K_j, \text{ and } j = 1,\dots, b$$

The problem is approximately linearized

$$\text{Minimize} \quad Z = \sum_{j \in L} Hf_j(\sum_{k} \lambda_{jk}Q_{jk})$$

subject to

$$\sum_{j=1}^{b} a_{ij}(\sum_{k} \lambda_{jk} Q_{jk}) = 0 \qquad i = 2,\dots, n$$

$$\sum_{j=1}^{b} b_{ij}(R_j(\sum_{k} \lambda_{jk} | Q_{jk} | Q_{jk}) - HN_j - Hf_j) = 0 \qquad i = 1,\dots, m$$

$$\sum_{k} \lambda_{jk} = 1 \qquad j = 1,\dots, b$$

$$\lambda_{jk} \geq 0 \qquad k = 0,\dots, K_j, \text{ and } j = 1,\dots, b$$

If the fan characteristic are approximated by filling a second–degree polynomial to known points $A_j + B_jQ_j + C_jQ_j^2$, where A_j, B_j and C_j are unique to a specific fan and setting, the problem becomes:

$$\text{Minimize} \quad Z = \sum_{j \in L} Q_j(A_j + B_jQ_j + C_jQ_j^2)$$

subject to

$$\sum_{j=1}^{b} a_{ij}Q_j = 0 \qquad i = 2,\dots, n$$

$$\sum_{j \notin L} b_{ij}R_j | Q_j | Q_j + \sum_{j \in L} b_{ij}(R_j | Q_j | Q_j - HN_j - (A_j+B_jQ_j+C_jQ_j^2))$$
$$= 0 \qquad i = 1,\dots, m$$

$$L_j \le Q_j \le U_j \qquad j = 1,\ldots, b$$

Using special ordered sets variables λ_{jk} for $j = 1,\ldots, b$ and following constraints,

$$Q_j = \sum_k \lambda_{jk} Q_{jk} \qquad j = 1,\ldots, b$$

$$\sum_k \lambda_{jk} = 1 \qquad j = 1,\ldots, b$$

$$\lambda_{jk} \ge 0 \qquad k = 0,\ldots, K_j, \text{ and } j = 1,\ldots, b$$

this problem can be linearized as follows:

$$\text{Minimize} \quad Z = \sum_{j \in L} \sum_k \lambda_{jk}(A_j Q_{jk} + B_j Q_{jk}^2 + C_j Q_{jk}^3)$$

subject to

$$\sum_{j=1}^{b} a_{ij}(\sum_k \lambda_{jk} Q_{jk}) = 0 \qquad i = 2,\ldots, n$$

$$\sum_{j \notin L} b_{ij}(R_j \sum_k \lambda_{jk} | Q_{jk} | Q_{jk} - HN_j) + \sum_{j \in L} b_{ij}\{\sum_k \lambda_{jk}[R_j | Q_{jk} | Q_{jk} - (A_j + B_j Q_{jk} + C_j Q_{jk}^2)] - HN_j\} = 0 \qquad i = 1,\ldots, m$$

$$\sum_k \lambda_{jk} = 1 \qquad j = 1,\ldots, b$$

$\lambda_{jk} \geq 0 \qquad\qquad k = 0,\ldots, K_j, \text{ and } j = 1,\ldots, b$

These problems can be solved by using the algorithm in Section 4.7.

5.3.2 Controlled Splitting Network Problem

In this problem, because flow quantity for each branch is fixed, the constraints for Kirchhoff's first law, already satisfied, will not show up in the model, and the problem becomes:

$$\text{Minimize } Z = \sum_{j\in L} (C_p + C_m)Y_j q_j HF_j + \sum_{j\in L} C_j Y_j$$

$$HF_j \begin{cases} = 0 & \text{if } Y_j = 0 \\ \geq HF_{jmin} & \text{if } Y_j = 1 \end{cases}$$

$$\sum_{j=1}^{b} b_{ij}(R_j|q_j|q_j + HR_j - HN_j - HF_j) = 0 \qquad i = 1,\ldots, m$$

$$HF_j,\ HR_j \geq 0$$

After defining the partial solution $\{Y_j\}$ and simplifying the model, the problem with respect to each of partial solutions becomes:

$$\text{Minimize } Z = \sum_{j\in N} q_j HF_j$$

$$HF_j \geq HF_{jmin} \qquad \text{for } j \in N$$

$$\sum_{j=1}^{b} b_{ij}(R_j q_j^2 + HR_j - HN_j - HF_j) = 0 \qquad i = 1,\ldots, m$$

$$HF_j, \ HR_j \geq 0$$

This is a linear program. It can be solved by only using the linear program part embodied in the algorithm. In this case, the special ordered sets and the branch and bound are unnecessary.

5.4 Computer Program

5.4.1 Introduction

A computer program has been developed based on the algorithm proposed in this research. This program is seen not only as a research aide, but as a possible tool for ventilation system planning and analysis.

For a given ventilation network, the program determines the optimal locations and sizes for all fans and regulators, and resulting air flow distribution corresponding to every partial solution. Then it compares these solutions based on the cost factors input and chooses the best solution among them.

This software uses the C, FORTRAN, and LINDO programming languages. The functions and relationship of these systems are illustrated in Figure 11.

The data input, model linearization, and main control programs are written in

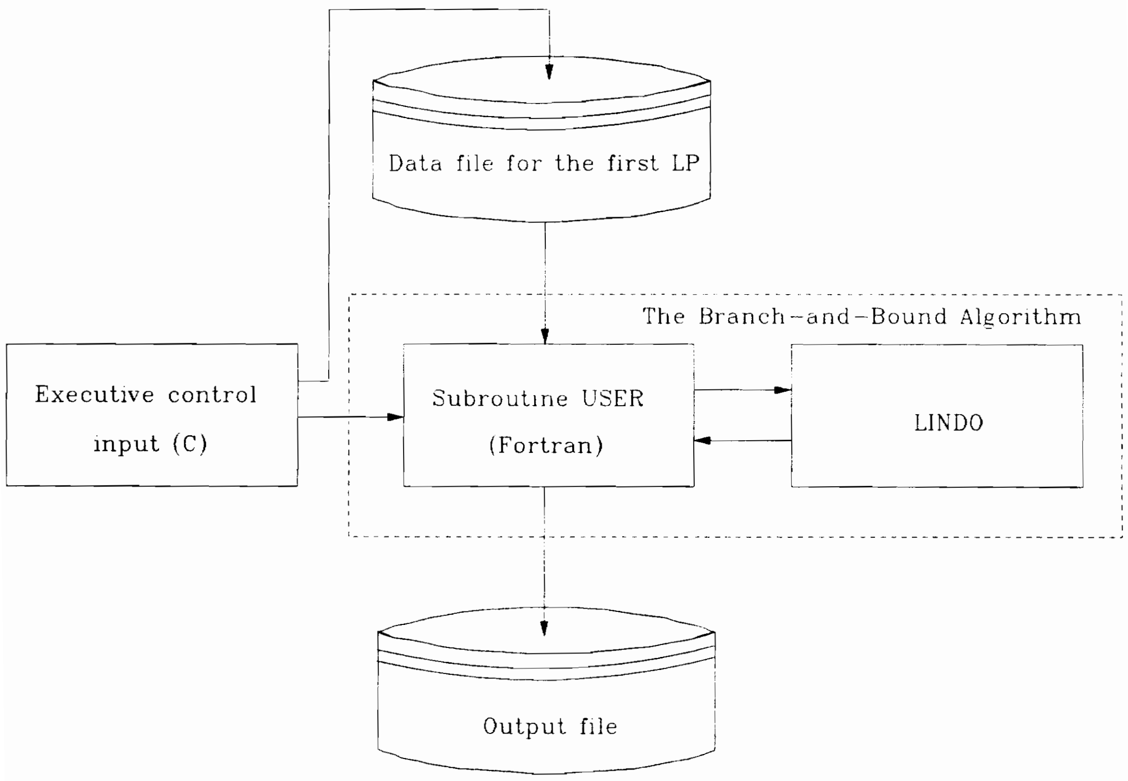

Data file for the first LP
The Branch-and-Bound Algorithm
Executive control input (C)
Subroutine USER (Fortran)
LINDO
Output file

Figure 11. Overview of the Model Configuration

C. The justification for selecting C is its superior compilation speed, well structured approach to programming, richness in library functions, and capacity to create a wide variety of complex data types. The C programs also are used to present the main menu, send the coefficients of all variables for first LP model and invoke the FORTRAN programs.

FORTRAN programs are used to formulate the first LP model, perform the modified branch and bound procedure, interface with LINDO, and write output reports. The programs, which run interactively, are invoked from C control programs. FORTRAN is utilized because of its computational capacity and ability to interact with LINDO.

The linearized programming model is solved using LINDO, a commercial linear programming package. LINDO provides the simplex solution and the USER subroutine is used to implement the modified branch and bound procedure. There is a file that serves as the input to LINDO. It contains all the coefficients for the objective function and constraints of the first linearized problem model, which is generated by C programs.

5.4.2 Program Development

The main part of the VENOPT (the computer program for OPTimization of VENtilation control device locations and sizes in underground mine ventilation systems) computer model is composed of 8 subroutines that are responsible for specific functions in the solution process. A detailed flow–chart of the model is given in in Figure 12. The associated program code is given in APPENDIX E.

The function 'new' in C allows user to create a data file in an interactive mode. The input data include both ventilation system parameters and branch data. Ventilation system parameters include the number of branches, number of nodes, and convergence factor. The branch data involve the initial node and final node, fan and regulator information, flow information, natural ventilation pressure, and resistance factor.

The function 'old' in C is used to read in data from the existing data file and review and modify them.

The function 'order' is used to identify the chords with respect to the network. The function 'mesh' is used to form the fundamental meshes of the network. The function 'form' prepares data matrix for Kirchhoffs' first and second laws. The function 'coef' calculates the coefficients for the first LP model.

The function 'store' is to store the coefficients of the first LP model variables produced by 'coef'. The function 'user' in C is used to call FORTRAN subroutine USER using the mixed–language interface capacity. It passes the data and the name of the file with the coefficients of the first LP variables to USER subroutine and invokes the FORTRAN programming.

The FORTRAN subroutine USER performs the modified branch and bound procedure. It includes: set up the first LP model, call LINDO to solve the current

LP problem, check the current solution status, determine which set to branch on , do automatic interpolation, place and take the problem from the alternative list, and compare the partial solutions and write solution report.

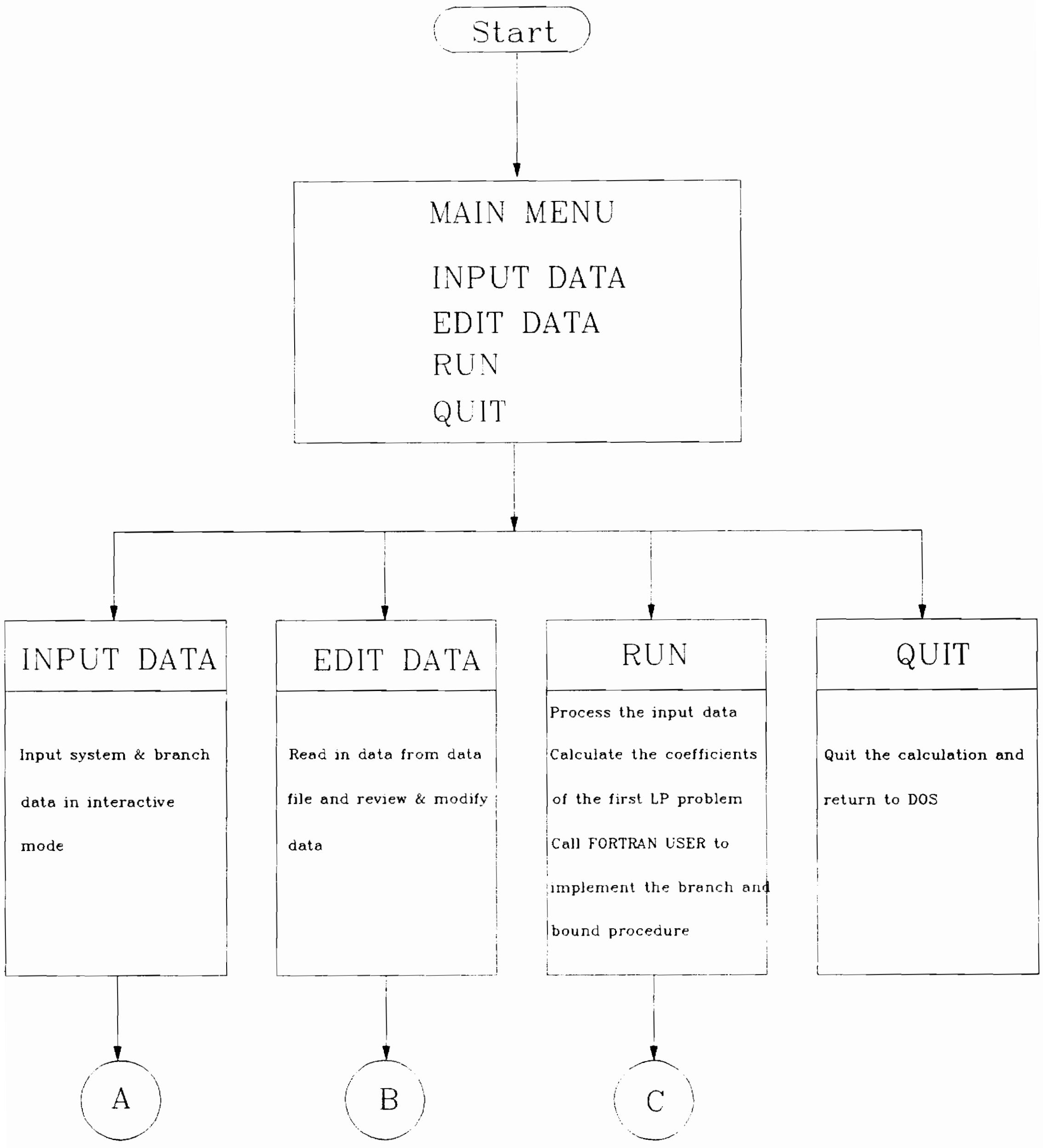

Start
MAIN MENU
INPUT DATA
EDIT DATA
RUN
QUIT
INPUT DATA
Input system & branch data in interactive mode
EDIT DATA
Read in data from data file and review & modify data
RUN
Process the input data
Calculate the coefficients of the first LP problem
Call FORTRAN USER to implement the branch and bound procedure
QUIT
Quit the calculation and return to DOS
A
B
C

Figure 12. Flow Chart of the Computer Program

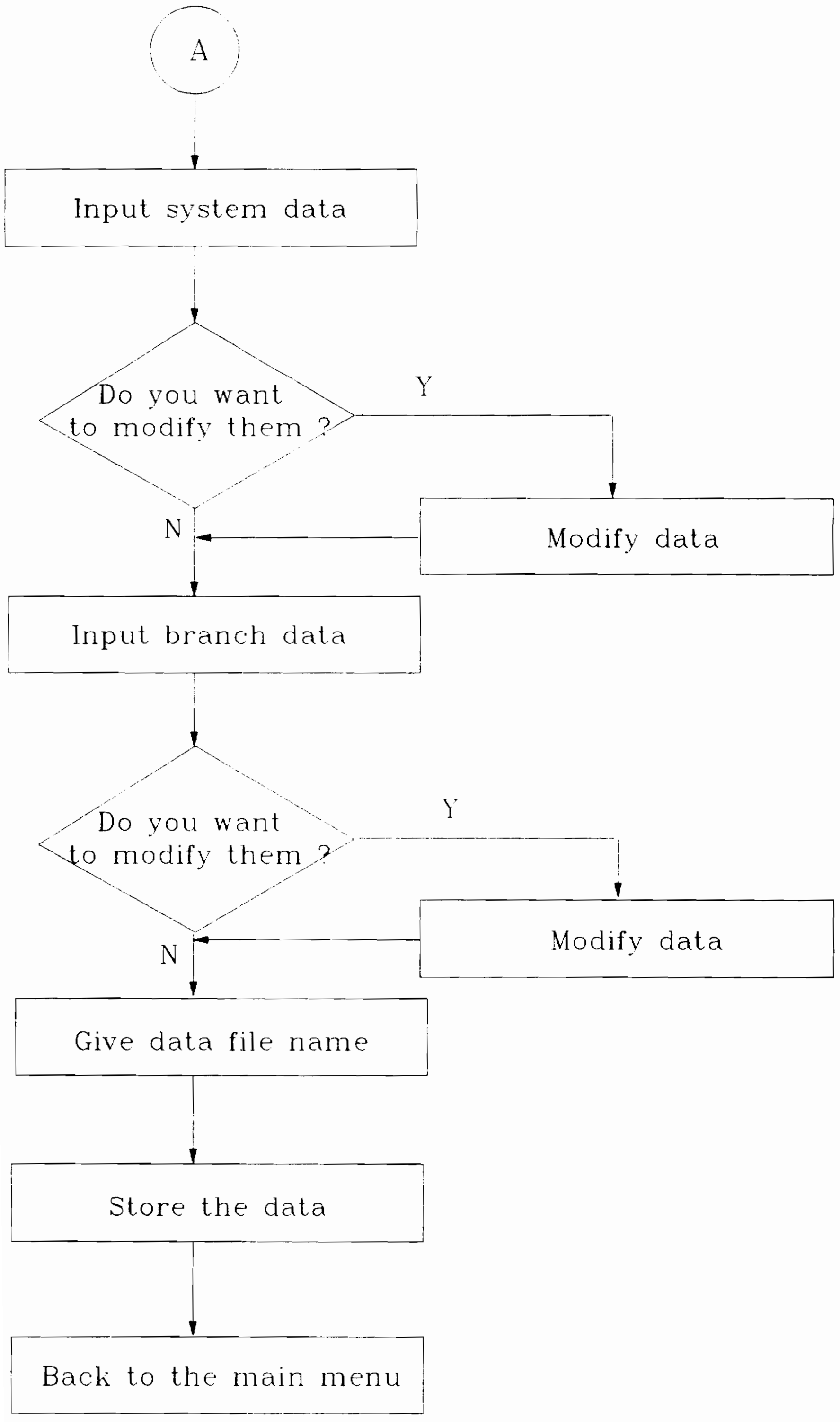


Figure 12-A. Flow Chart of the Computer Program

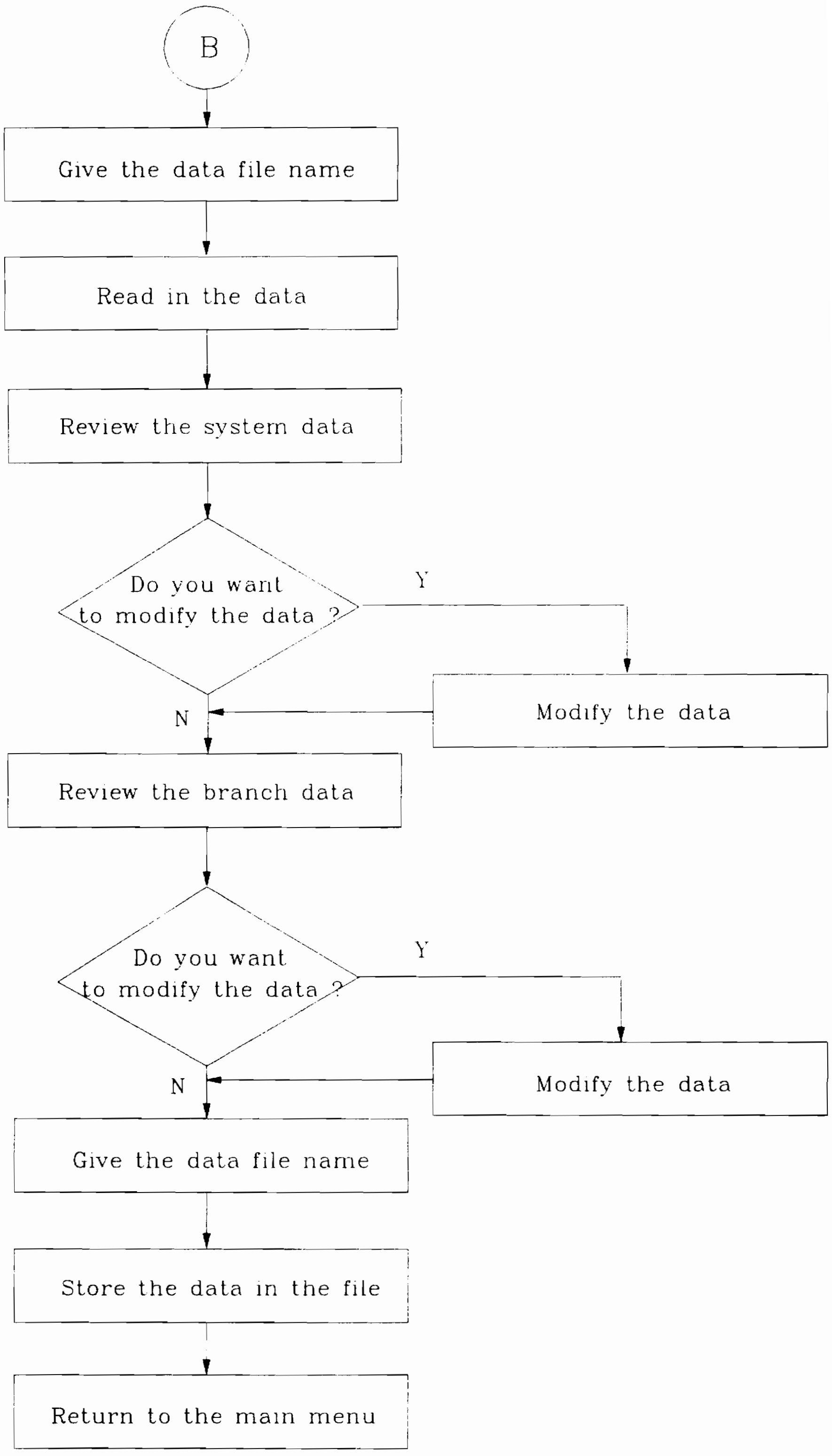


Fig 12-B. Flow Chart of the Computer Program

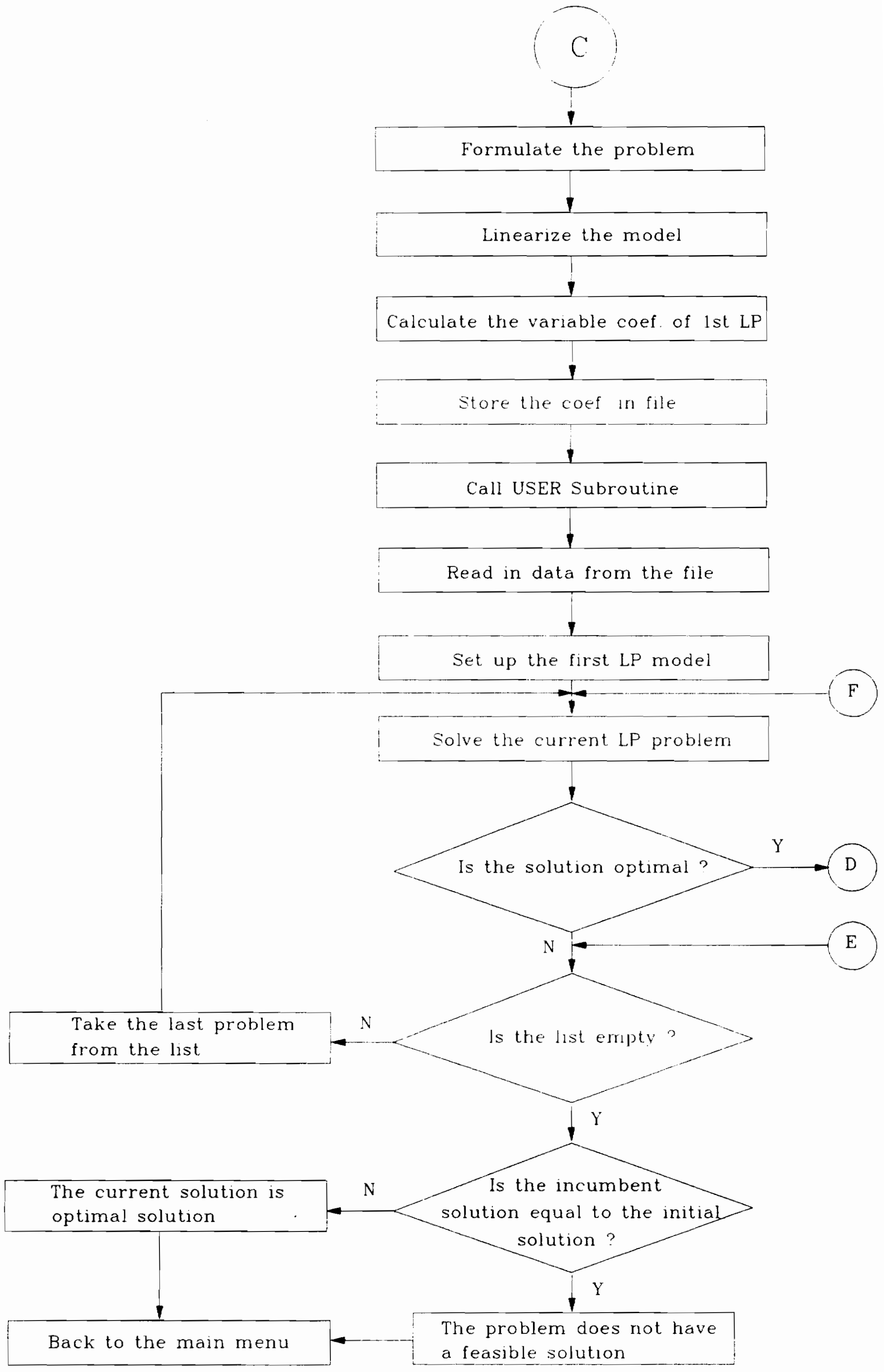
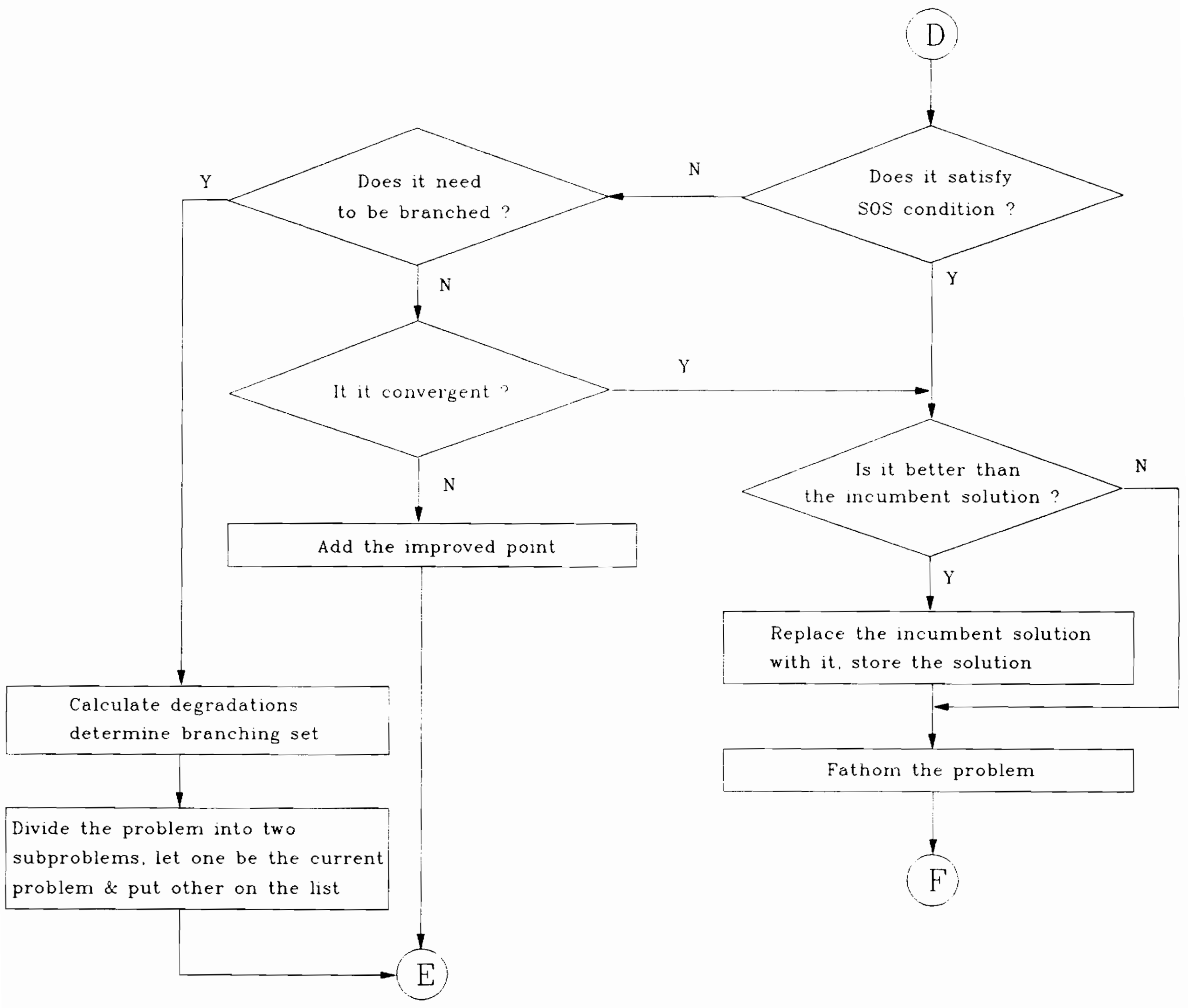

C
Formulate the problem
Linearize the model
Calculate the variable coef. of 1st LP
Store the coef. in file
Call USER Subroutine
Read in data from the file
Set up the first LP model
F
Solve the current LP problem
Is the solution optimal ?
Y
D
N
E
Take the last problem from the list
N
Is the list empty ?
Y
The current solution is optimal solution
N
Is the incumbent solution equal to the initial solution ?
Y
Back to the main menu
The problem does not have a feasible solution

Figure 12-C. Flow Chart of the Computer Program



D
Does it satisfy SOS condition ?
N
Y
Does it need to be branched ?
Y
N
It it convergent ?
Y
N
Add the improved point
Is it better than the incumbent solution ?
N
Y
Replace the incumbent solution with it, store the solution
Fathom the problem
F
Calculate degradations determine branching set
Divide the problem into two subproblems, let one be the current problem & put other on the list
E

Figure 12-D. Flow Chart of the Computer Program

CHAPTER VI APPLICATIONS OF METHODOLOGY

6.1 Introduction

In this chapter an example ventilation network is used to validate the ventilation network optimization model developed. As shown in Figure 13, this network consists of 12 branches and 8 nodes. There are only two fixed–flow branches, branches 1 and 6, as working branches. The problem is to determine optimal location and size of ventilation control devices and flow distribution in other branches. Obviously, it is a semi–controlled splitting or generalized network problem. According to the technical and production conditions, it is assumed that fans could be located in branches 3, 4, 10, and 12 and regulators could be located in branches 6, 8, and 9. The resistance of every branch is given. These input data are given in Table I. It is further assumed that one fan will always be located in branch 12 which corresponds to a surface fan. As a result, all possible combinations of four fans may be listed as:

(1) Fan is allowed in branch 12 i.e., $Y_{12} = 1$

(2) Fans are allowed in branches 3 and 12 i.e., $Y_3 = Y_{12} = 1$

Fans are allowed in branches 4 and 12 i.e., $Y_4 = Y_{12} = 1$

Fans are allowed in branches 10 and 12 i.e., $Y_{10} = Y_{12} = 1$

(3) Fans are allowed in branches 3, 4, and 12 i.e., $Y_3 = Y_4 = Y_{12} = 1$

Fans are allowed in branches 3, 10, and 12 i.e., $Y_3 = Y_{10} = Y_{12} = 1$

Fans are allowed in branches 3, 10, and 12 i.e., $Y_4 = Y_{10} = Y_{12} = 1$

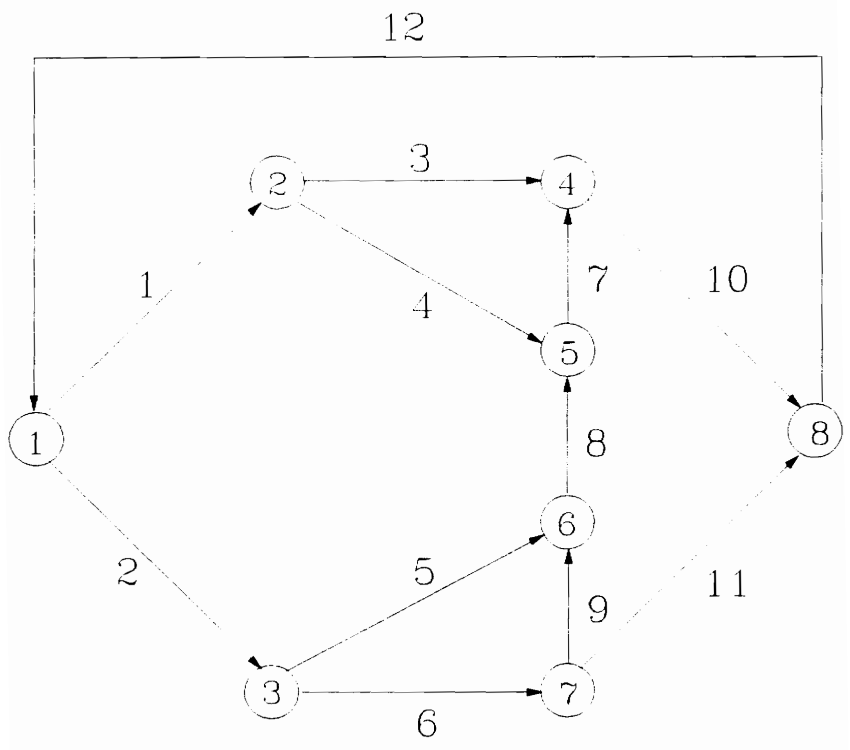


Figure 13. Example Ventilation Network

Table I. Input Data

Branc h no.	Resistance $N.s^2/m^8$	Flow rate m^3/s	Fan allowed	Regulator allowed
1	0.60	50	no	no
2	0.03		no	no
3	0.25		yes	no
4	0.45		yes	no
5	0.50		no	no
6	0.16	50	no	yes
7	0.04		no	no
8	0.01		no	yes
9	0.10		no	yes
10	0.02		yes	no
11	0.88		no	no
12	0.00		yes	no

6.2 Partial Solution (1)

Corresponding to Partial Solution (1): fans in Branch 12, the algorithm selects the spanning tree {2, 3, 5, 7, 9, 10, 11} and chords {1, 4, 6, 8, 12} (see Fig. 14) and forms the fundamental meshes as:

Mesh 1: 1, –2, –5, 9, –11, 10, 3

Mesh 2: 4, 7, –3

Mesh 3: 6, 9, –5

Mesh 4: 8, 9, –11, 10, 7

Mesh 5: 12, 2, 5, –9, 11,

The problem is formulated as following:

Minimize $Q_{12}HF_{12}$

subject to

$Q_{12}HF_{12} \geq P_{12}$

$Q_3 + Q_4 = q_1$

$Q_2 - Q_5 = q_6$

$-Q_3 - Q_7 + Q_{10} = 0$

$-Q_4 + Q_7 - Q_8 = 0$

$-Q_5 + Q_8 - Q_9 = 0$

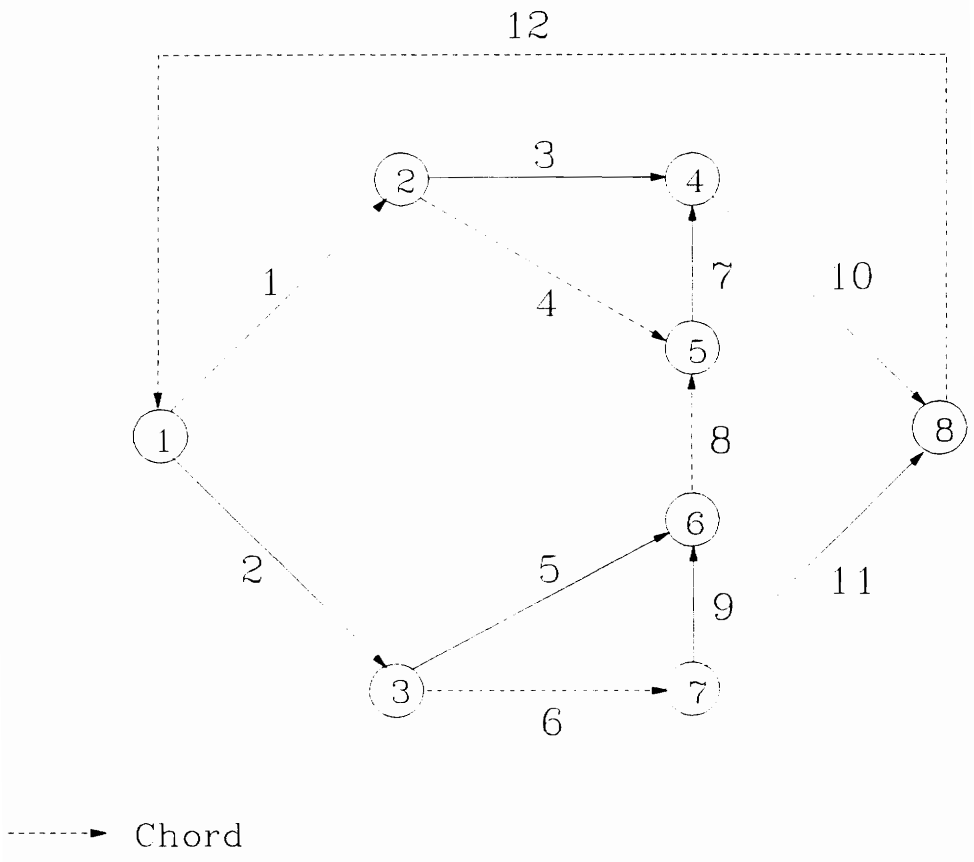


Figure 14. The Spanning Tree for the Partial Solution (1)

$Q_9 + Q_{11} = q_6$

$-Q_{10} - Q_{11} + Q_{12} = 0$

$R_1q_1^2 - R_2Q_2^2 + R_3Q_3^2 - R_5Q_5^2 + R_9Q_9^2 + HR_9 + R_{10}Q_{10}^2 - R_{11}Q_{11}^2 = 0$

$-R_3Q_3^2 + R_4Q_4^2 + R_7Q_7^2 = 0$

$-R_5Q_5^2 + R_6Q_6^2 + HR_6 + R_9Q_9^2 + HR_9 = 0$

$R_7Q_7^2 + R_8Q_8^2 + HR_8 + R_9Q_9^2 + HR_9 + R_{10}Q_{10}^2 - R_{11}Q_{11}^2 = 0$

$R_2Q_2^2 + R_5Q_5^2 - R_9Q_9^2 - HR_9 + R_{11}Q_{11}^2 - HF_{12} = 0$

$L_2 \leq Q_2 \leq U_2$

$L_3 \leq Q_3 \leq U_3$

$L_4 \leq Q_4 \leq U_4$

$L_5 \leq Q_5 \leq U_5$

$L_7 \leq Q_7 \leq U_7$

$L_8 \leq Q_8 \leq U_8$

$L_9 \leq Q_9 \leq U_9$

$L_{10} \leq Q_{10} \leq U_{10}$

$L_{11} \leq Q_{11} \leq U_{11}$

$L_{12} \leq Q_{12} \leq U_{12}$

$HR_6, HR_8, HR_9 \geq 0$

This formulation can be simplified as follows if flow quantities of the tree branches are represented by the chords:

Minimize $Q_{12}HF_{12}$

subject to

$$Q_{12}HF_{12} \geq P_{12}$$

$$Q_4 - Q_7 + Q_8 = 0$$
$$-Q_8 + Q_9 + Q_{12} = q_1 + q_6$$
$$-Q_8 - Q_{11} + Q_{12} = q_1$$

$$2R_3q_1Q_4 - R_3Q_4^2 - 2R_{10}q_2Q_8 - R_{10}Q_8^2 - R_9Q_9^2 - HR_9 + R_{11}Q_{11}^2$$
$$- 2[R_2q_1 + R_5(q_1 + q_6)]Q_{12} - (R_2 + R_5)Q_{12}^2$$
$$= (R_1 - R_2 + R_3 + R_{10})q_1^2 - R_5(q_1 + q_6)^2$$
$$2R_3q_1Q_4 + (R_4 - R_3)Q_4^2 + R_7Q_7^2 = R_3q_1^2$$
$$-R_9Q_9^2 + 2R_5(q_1+q_6)Q_{12} - R_5Q_{12}^2 - HR_6 - HR_9 = R_6q_6^2+R_5(q_1+q_6)^2$$
$$-R_7Q_7^2 - 2R_{10}q_1Q_8 - (R_8+R_{10})Q_8^2 - R_9Q_9^2 + R_{11}Q_{11}^2 - HR_8 - HR_9 = R_{10}q_1^2$$
$$R_9Q_9^2 - R_{11}Q_{11}^2+ 2[R_2q_1 + R_5(q_1+q_6)]Q_{12} - (R_2+R_5)Q_{12}^2$$
$$+ HR_9 + HF_{12} = R_2q_1^2 + R_5(q_1+q_6)^2$$

$$L_4 \leq Q_4 \leq U_4$$
$$L_7 \leq Q_7 \leq U_7$$
$$L_8 \leq Q_8 \leq U_8$$
$$L_9 \leq Q_9 \leq U_9$$
$$L_{11} \leq Q_{11} \leq U_{11}$$
$$L_{12} \leq Q_{12} \leq U_{12}$$

$HR_6, HR_8, HR_9 \geq 0$

where $Q_2 = Q_{12} - q_1$, $Q_3 = q_1 - Q_4$, $Q_5 = Q_{12} - (q_1+q_6)$, $Q_{10} = Q_8 + q_1$.

We introduce special ordered set variables u_k, v_k, w_k, x_k, y_k for Q_4, Q_7, Q_8, Q_9, and Q_{11} and define following equations:

$$Q_4 = \sum_k Q_{4k}u_k, \qquad \sum_k u_k = 1 \qquad u_k \geq 0$$

$$Q_7 = \sum_k Q_{7k}v_k, \qquad \sum_k v_k = 1 \qquad v_k \geq 0$$

$$Q_8 = \sum_k Q_{8k}w_k, \qquad \sum_k w_k = 1 \qquad w_k \geq 0$$

$$Q_9 = \sum_k Q_{9k}x_k, \qquad \sum_k x_k = 1 \qquad x_k \geq 0$$

$$Q_{11} = \sum_k Q_{11k}y_k, \qquad \sum_k y_k = 1 \qquad y_k \geq 0$$

For the bilinear term $Q_{12}HF_{12}$, in objective function, two set of variables z_{1k} and z_{2k} and following equations are introduced:

$$\sum_k z_{1k} + \sum_k z_{2k} = 1, \qquad z_{1k}, z_{2k} \geq 0$$

$$Q_{12} = \sum_k Q_{12k}z_{1k} + \sum_k Q_{12k}z_{2k}$$

$$HF_{12} = \sum_k HF_{12min}z_{1k} + \sum_k HF_{12max}z_{2k}$$

$$Q_{12}HF_{12} = \sum_k HF_{12min}Q_{12k}z_{1k} + \sum_k HF_{12max}Q_{12k}z_{2k}$$

The problem is linearized as follows:

Minimize $\sum_k HF_{12min}Q_{12k}z_{1k} + \sum_k HF_{12max}Q_{12k}z_{2k}$

subject to

$$\sum_k Q_{4k}u_k - \sum_k Q_{7k}v_k + \sum_k Q_{8k}w_k = 0$$

$$-\sum_k Q_{8k}w_k + \sum_k Q_{9k}x_k + \sum_k Q_{12k}z_{1k} + \sum_k Q_{12k}z_{2k} = q_1 + q_6$$

$$-\sum_k Q_{8k}w_k + \sum_k Q_{11k}y_k + \sum_k Q_{12k}z_{1k} + \sum_k Q_{12k}z_{2k} = q_1$$

$$R_3\sum_k(2q_1Q_{4k} - Q_{4k}^2)u_k - R_{10}\sum_k (2Q_{8k} + Q_{8k}^2)w_k - R_9\sum_k Q_{9k}^2x_k - HR_9$$
$$+ R_{11}\sum_k Q_{11k}^2y_k - \sum_k \{2[R_2q_1+R_5(q_1+q_6)]Q_{12k}-(R_2+R_5)Q_{12k}^2\}z_{1k}$$
$$- \sum_k\{2[R_2q_1+R_5(q_1+q_6)]Q_{12k}-(R_2+R_5)Q_{12k}^2\}z_{2k}$$
$$= (R_1 - R_2 + R_3 + R_{10})q_1^2 - R_5(q_1 + q_6)^2$$

$$\sum_k[2R_3q_1Q_{4k} + (R_4-R_3)Q_{4k}^2]u_k + R_7\sum_k Q_{7k}^2v_k = R_3q_1^2$$

$$-R_9\sum_k Q_{9k}^2x_k - HR_6 - HR_9 + R_5\sum_k [2(q_1+q_6)Q_{12k} - Q_{12k}^2]z_{1k}$$
$$+ R_5\sum_k [2(q_1+q_6)Q_{12k} - Q_{12k}^2]z_{2k} = R_6q_6^2 + R_5(q_1+q_6)^2$$

$$-R_7\sum_k Q_{7k}^2v_k - \sum_k[2R_{10}q_1Q_{8k} + (R_8+R_{10})Q_{8k}^2]w_k - R_9\sum_k Q_{9k}^2x_k$$
$$+ R_{11}\sum_k Q_{11k}^2y_k - HR_8 - HR_9 = R_{10}q_1^2$$

$$R_9\sum_k Q_{9k}^2x_k - R_{11}\sum_k Q_{11k}^2y_k + \sum_k \{2[R_2q_1 + R_5(q_1+q_6)]Q_{12k} - (R_2+R_5)Q_{12k}^2\}z_{1k}$$
$$+ \sum_k \{2[R_2q_1 + R_5(q_1+q_6)]Q_{12k} - (R_2+R_5)Q_{12k}^2\}z_{2k} + HR_9 + HF_{12}$$
$$= R_2q_1^2 + R_5(q_1+q_6)^2$$

$$\sum_k u_k = 1$$

$$\sum_k v_k = 1$$

$$\sum_k w_k = 1$$

$$\sum_k x_k = 1$$

$$\sum_k y_k = 1$$

$$\sum_k z_{1k} + \sum_k z_{2k} = 1$$

$$\sum_k HF_{12min} z_{1k} + \sum_k HF_{12max} z_{2k} - HF_{12} = 0$$

$$u_k,\ v_k,\ w_k,\ x_k,\ y_k,\ z_{1k},\ z_{2k} \geq 0$$

$$HR_6,\ HR_8,\ HR_9 \geq 0$$

Using the optimization algorithm developed, the optimal solution can be found out and the results are given in Table II. They indicate that the optimal solution is to site the main fan in branch 12 and a regulator in branch 8.

6.3 Partial Solution (2)

There are three alternatives: fans in Branches 3 and 12, fans in Branches 4 and 12, and fans in Branches 10 and 12. Corresponding to the partial solution of fans in branches 10 and 12, the algorithm chooses the spanning tree {2, 3, 5, 7, 8, 9, 11} (see Fig. 15) and the problem can be formulated as following:

Table II. Algorithm Output For Partial Solution of Fan in Branch 12

Branch #	Q_j (m^3/s)	HF_j (Pa.)	HR_j (Pa.)
1	50.00		
2	78.69		
3	32.62		
4	17.38		
5	28.69		
6	50.00		
7	57.02		
8	39.65		1022
9	10.96		
10	89.65		
11	39.04		
12	128.69	1927	

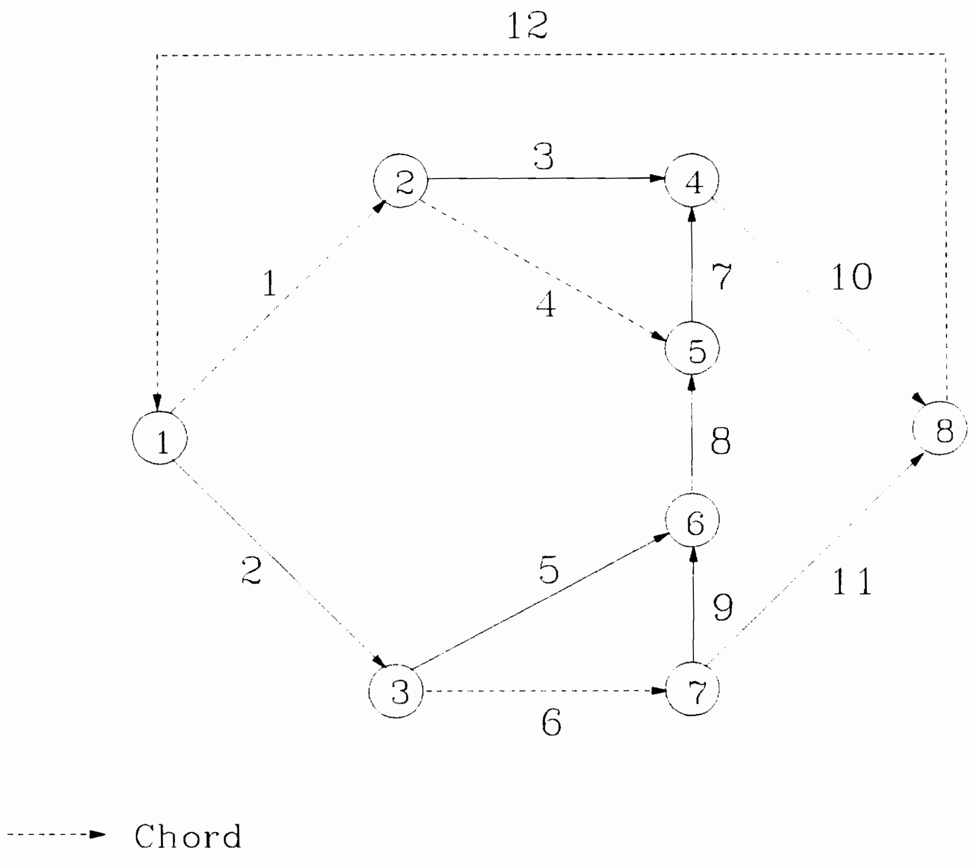


Figure 15. The Spanning Tree for the Partial Solution (2)

Minimize $\quad Q_{10}HF_{10} + Q_{12}HF_{12}$

subject to

$$Q_{10}HF_{10} \geq P_{10}$$
$$Q_{12}HF_{12} \geq P_{12}$$

$$Q_4 - Q_7 + Q_{10} = q_1$$
$$Q_9 - Q_{10} + Q_{12} = q_6$$
$$Q_{10} + Q_{11} - Q_{12} = 0$$

$$-2R_3q_1Q_4 + R_3Q_4^2 - R_7Q_7^2 + 2R_8q_1Q_{10} - R_8Q_{10}^2 - HR_8$$
$$+ 2[R_2q_1 + R_5(q_1 + q_6)]Q_{12} - (R_2 + R_5)Q_{12}^2$$
$$= (R_2 + R_8 - R_1 - R_3)q_1^2 + R_5(q_1 + q_6)^2$$
$$2R_3q_1Q_4 + (R_4 - R_3)Q_4^2 + R_7Q_7^2 = R_3q_1^2$$
$$HR_6 + R_9Q_9^2 + HR_9 + 2R_5(q_1+q_6)Q_{12} - R_5Q_{12}^2 = R_5(q_1+q_6)^2 - R_6q_6^2$$
$$-R_7Q_7^2 - HR_8 - R_9Q_9^2 - HR_9 + 2R_8q_1Q_{10} - (R_8+R_{10})Q_{10}^2$$
$$+ HF_{10} + R_{11}Q_{11}^2 \quad = R_8q_1^2$$
$$R_9Q_9^2 + HR_9 - R_{11}Q_{11}^2 + HF_{12} + 2[R_2q_1 + R_5(q_1+q_6)]Q_{12} - (R_2+R_5)Q_{12}^2$$
$$= R_2q_1^2 + R_5(q_1+q_6)^2$$

$$L_4 \leq Q_4 \leq U_4$$
$$L_7 \leq Q_7 \leq U_7$$
$$L_9 \leq Q_9 \leq U_9$$
$$L_{10} \leq Q_{10} \leq U_{10}$$

$$L_{11} \leq Q_{11} \leq U_{11}$$

$$L_{12} \leq Q_{12} \leq U_{12}$$

$$HR_6,\ HR_8,\ HR_9 \geq 0$$

where $Q_2 = Q_{12} - q_1$, $Q_3 = q_1 - Q_4$, $Q_5 = Q_{12} - (q_1+q_6)$, $Q_8 = Q_{10} - q_1$.

Use special ordered set variables u_k, v_k, w_k, and x_k and following constraints for Q_4, Q_7, Q_9, and Q_{11}.

$$Q_4 = \sum_k Q_{4k}u_k, \qquad \sum_k u_k = 1, \qquad u_k \geq 0$$

$$Q_7 = \sum_k Q_{7k}v_k, \qquad \sum_k v_k = 1, \qquad v_k \geq 0$$

$$Q_9 = \sum_k Q_{9k}w_k, \qquad \sum_k w_k = 1, \qquad w_k \geq 0$$

$$Q_{11} = \sum_k Q_{11k}x_k, \qquad \sum_k x_k = 1, \qquad x_k \geq 0$$

Use four sets of variables y_{1k}, y_{2k}, z_{1k}, and z_{2k} for the product terms $Q_{10}HF_{10}$ and $Q_{12}HF_{12}$ and define the following equations:

$$\sum_k y_{1k} + \sum_k y_{2k} = 1 \qquad\qquad y_{1k},\ y_{2k} \geq 0$$

$$Q_{10} = \sum_k Q_{10k}y_{1k} + \sum_k Q_{10k}y_{2k}$$

$$HF_{10} = \sum_k HF_{10min}y_{1k} + \sum_k HF_{10max}y_{2k}$$

$$Q_{10}HF_{10} = \sum_k HF_{10min}Q_{10k}y_{1k} + \sum_k HF_{10max}Q_{10k}y_{2k}$$

$$\sum_k z_{1k} + \sum_k z_{2k} = 1 \qquad\qquad z_{1k},\ z_{2k} \geq 0$$

$$Q_{12} = \sum_k Q_{12k}z_{1k} + \sum_k Q_{12k}z_{2k}$$

$$HF_{12} = \sum_k HF_{12min}z_{1k} + \sum_k HF_{12max}z_{2k}$$

$$Q_{12}HF_{12} = \sum_k HF_{12min}Q_{12k}z_{1k} + \sum_k HF_{12max}Q_{12k}z_{2k}$$

Therefore, the problem can be linearized as follows:

$$\text{Minimize} \quad \sum_k HF_{10min}Q_{10k}y_{1k} + \sum_k HF_{10max}Q_{10k}y_{2k}$$

$$\sum_k HF_{12min}Q_{12k}z_{1k} + \sum_k HF_{12max}Q_{12k}z_{2k}$$

subject to

$$\sum_k Q_{4k}u_k - \sum_k Q_{7k}v_k + \sum_k Q_{10k}y_{1k} + \sum_k Q_{10k}y_{2k} = q_1$$

$$\sum_k Q_{9k}w_k - \sum_k Q_{10k}y_{1k} - \sum_k Q_{10k}y_{2k} + \sum_k Q_{12k}z_{1k} + \sum_k Q_{12k}z_{2k} = q_6$$

$$\sum_k Q_{10k}y_{1k} + \sum_k Q_{10k}y_{2k} + \sum_k Q_{11k}x_k - \sum_k Q_{12k}z_{1k} - \sum_k Q_{12k}z_{2k} = 0$$

$$R_3\sum_k(-2q_1Q_{4k} + Q_{4k}^2)u_k - R_7\sum_k Q_{7k}^2v_k + R_8\sum_k (2q_1Q_{10k} - Q_{10k}^2)y_{1k}$$

$$+ R_8\sum_k (2q_1Q_{10k} - Q_{10k}^2)y_{2k} + \sum_k\{2[R_2q_1 + R_5(q_1+q_6)]Q_{12k}$$

$$- (R_2+R_5)Q_{12k}^2\}z_{1k} + \sum_k\{2[R_2q_1+R_5(q_1+q_6)]Q_{12k} - (R_2+R_5)Q_{12k}^2\}z_{2k}$$

$$= (R_2+R_8-R_1-R_3)q_1^2 - R_5(q_1+q_6)^2$$

$$\sum_k[2R_3q_1Q_{4k} + (R_4-R_3)Q_{4k}^2]u_k + R_7\sum_k Q_{7k}^2v_k = R_3q_1^2$$

$$R_9\sum_k Q_{9k}^2w_k + HR_6 + HR_9 + R_5\sum_k [2(q_1+q_6)Q_{12k} - Q_{12k}^2]z_{1k}$$

$$+ R_5\sum_k [2(q_1+q_6)Q_{12k} - Q_{12k}^2]z_{2k} = R_5(q_1+q_6)^2 - R_6q_6^2$$

$$-R_7\sum_k Q_{7k}^2 v_k - HR_8 - R_9\sum_k Q_{9k}^2 w_k - HR_9 + \sum_k[2R_8 q_1 Q_{10k} - (R_8+R_{10})Q_{10k}^2]y_{1k}$$

$$+ \sum_k[2R_8 q_1 Q_{10k} - (R_8+R_{10})Q_{10k}^2]y_{2k} + R_{11}\sum_k Q_{11k}^2 x_k + HF_{10} = R_8 q_1^2$$

$$R_9\sum_k Q_{9k}^2 w_k + HR_9 - R_{11}\sum_k Q_{11k}^2 x_k + \sum_k \{2[R_2 q_1 - R_5(q_1+q_6)]Q_{12k}$$

$$+ (R_2+R_5)Q_{12k}^2\}z_{1k} + \sum_k \{2[R_2 q_1 + R_5(q_1+q_6)]Q_{12k} - (R_2+R_5)Q_{12k}^2\}z_{2k}$$

$$+ HF_{12} = R_2 q_1^2 + R_5(q_1+q_6)^2$$

$$\sum_k u_k = 1$$

$$\sum_k v_k = 1$$

$$\sum_k w_k = 1$$

$$\sum_k x_k = 1$$

$$\sum_k y_{1k} + \sum_k y_{2k} = 1$$

$$\sum_k z_{1k} + \sum_k z_{2k} = 1$$

$$\sum_k HF_{10min} y_{1k} + \sum_k HF_{10max} y_{2k} - HF_{10} = 0$$

$$\sum_k HF_{12min} z_{1k} + \sum_k HF_{12max} z_{2k} - HF_{12} = 0$$

$$u_k,\ v_k,\ w_k,\ x_k,\ y_{1k},\ y_{2k},\ z_{1k},\ z_{2k} \geq 0$$

$$HR_6,\ HR_8,\ HR_9 \geq 0$$

The algorithm output is given in Table III. Similarly, we can solve other two alternatives. Table IV shows the algorithm output for partial solution of fans in Branches 3 and 12. Table V summarizes the results of partial solution in which fans

Table III. Algorithm Output For Partial Solution of Fans in Branches 10 and 12

Branch #	Q_j (m^3/s)	HF_j (Pa.)	HR_j (Pa.)
1	50.00		
2	79.65		
3	33.94		
4	16.06		
5	29.65		
6	50.00		
7	65.56		
8	49.50		962
9	19.86		
10	99.50	596	
11	30.15		
12	129.65	1390	

Table IV. Algorithm Output For Partial Solution of Fans in Branches 3 and 12

Branch #	Q_j (m^3/s)	HF_j (Pa.)	HR_j (Pa.)
1	50.00		
2	78.83		
3	39.55	235	
4	10.45		
5	28.83		
6	50.00		
7	51.79		
8	41.34		930
9	12.51		
10	91.34		
11	37.49		
12	128.83	1823	

Table V. Algorithm Output For Partial Solution of Fans in Branches 4 and 12

Branch #	Q_j (m^3/s)	HF_j (Pa.)	HR_j (Pa.)
1	50.00		
2	78.85		
3	24.01		
4	25.99	342	
5	28.85		
6	50.00		
7	67.53		
8	41.53		842
9	12.69		
10	91.54		
11	37.31		
12	128.85	1812	

are allowed in Branches 4 and 12.

6.4 Partial Solution (3)

There are 3 possible combinations in the partial solution. Corresponding to the partial solution of fans in Branches 3, 4, and 12, the algorithm chooses the spanning tree {1, 2, 5, 7, 8, 9, 10} as shown in Fig. 16 and the problem can be formulated as following:

$$\text{Minimize} \quad Q_3HF_3 + Q_4HF_4 + Q_{12}HF_{12}$$

subject to

$$Q_3HF_3 \geq P_3$$
$$Q_4HF_4 \geq P_4$$
$$Q_{12}HF_{12} \geq P_{12}$$

$$Q_3 + Q_4 = q_1$$
$$Q_4 - Q_7 - Q_{11} + Q_{12} = q_1$$
$$-Q_8 - Q_{11} + Q_{12} = q_1$$

$$R_3Q_3^2 - R_7Q_7^2 - R_8Q_8^2 + 2[R_2q_1 + R_5(q_1+q_6)]Q_{12} - (R_2+R_5)Q_{12}^2$$
$$- HR_8 - HF_3 = (R_2-R_1)q_1^2 + R_5(q_1+q_6)^2$$
$$R_4Q_4^2 - R_8Q_8^2 + 2[R_2q_1 + R_5(q_1+q_6)]Q_{12} - (R_2+R_5)Q_{12}^2$$
$$- HR_8 - HF_4 = (R_2-R_1)q_1^2 + R_5(q_1+q_6)^2$$
$$2R_9q_6Q_{11} - R_9Q_{11}^2 - 2R_5(q_1+q_6)Q_{12} + R_5Q_{12}^2$$

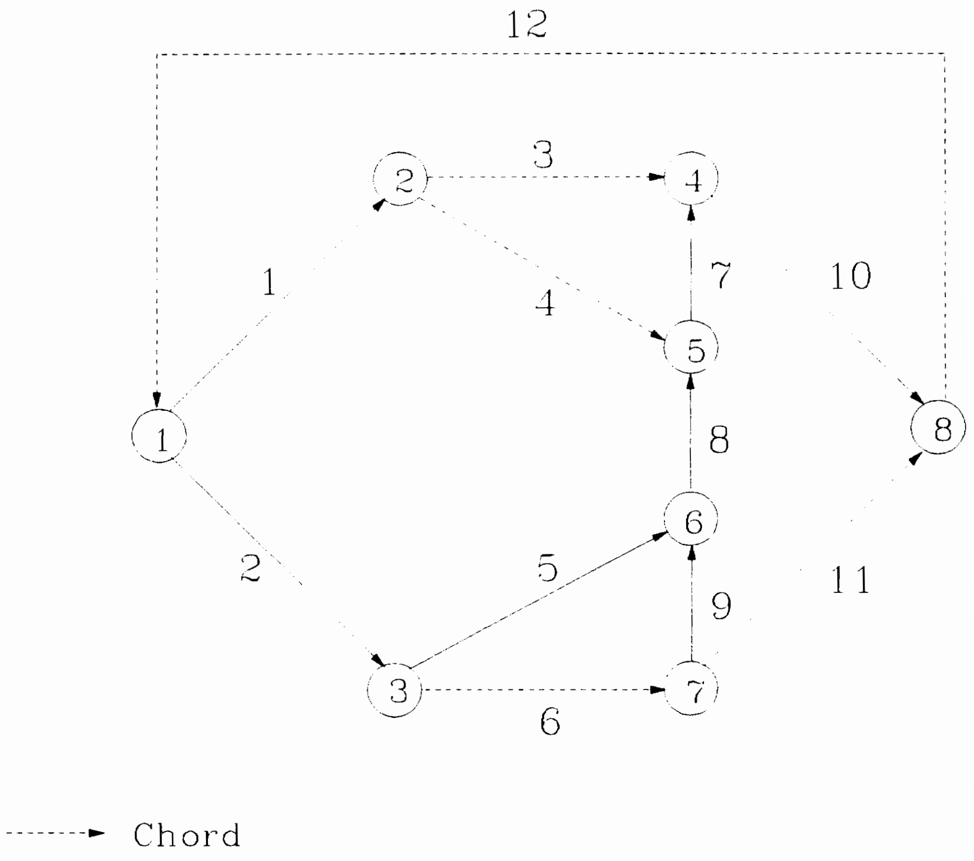


Figure 16. The Spanning Tree for the Partial Solution (3)

$$- HR_6 - HR_9 = (R_9+R_9)q_6^2 - R_5(q_1 + q_6)^2$$

$$-R_7Q_7^2 - R_8Q_8^2 - R_{10}Q_{10}^2 + 2R_9q_6Q_{11} + (R_{11} - R_9)Q_{11}^2$$

$$- HR_8 - HR_9 = R_9q_6^2$$

$$-R_7Q_7^2 - 2R_{10}q_1Q_8 - (R_8+R_{10}) + 2[R_2q_1 + R_5(q_1+q_6)]Q_{12} - (R_2+R_5)Q_{12}^2$$

$$- HR_8 + HF_{12} = (R_2+R_{10})q_1^2 + R_5(q_1+q_6)^2$$

$L_3 \leq Q_3 \leq U_3$

$L_4 \leq Q_4 \leq U_4$

$L_7 \leq Q_7 \leq U_7$

$L_8 \leq Q_8 \leq U_8$

$L_{11} \leq Q_{11} \leq U_{11}$

$L_{12} \leq Q_{12} \leq U_{12}$

$HR_6, HR_8, HR_9 \geq 0$

where $Q_2 = Q_{12} - q_1$, $Q_5 = Q_{12} - (q_1+q_6)$, $Q_9 = q_6 - Q_{11}$, $Q_{10} = Q_8 + q_1$

Use special ordered set variables u_k, v_k, w_k and following constraints for Q_7, Q_8, and Q_{11}.

$$Q_7 = \sum_k Q_{7k}u_k, \qquad \sum_k u_k = 1, \qquad u_k \geq 0$$

$$Q_8 = \sum_k Q_{8k}v_k, \qquad \sum_k v_k = 1, \qquad v_k \geq 0$$

$$Q_{11} = \sum_k Q_{11k}w_k, \qquad \sum_k w_k = 1, \qquad w_k \geq 0$$

Use six sets of variables x_{1k}, x_{2k}, y_{1k}, y_{2k}, z_{1k}, and z_{2k} for the product terms Q_3HF_3, Q_4HF_4, and $Q_{12}HF_{12}$ and define the following equations:

$$\sum_k x_{1k} + \sum_k x_{2k} = 1, \qquad x_{1k}, x_{2k} \geq 0$$

$$Q_3 = \sum_k Q_{3k}x_{1k} + \sum_k Q_{3k}x_{2k}$$

$$HF_3 = \sum_k HF_{3min}x_{1k} + \sum_k HF_{3max}x_{2k}$$

$$Q_3HF_3 = \sum_k HF_{3min}Q_{3k}x_{1k} + \sum_k HF_{3max}Q_{3k}x_{2k}$$

$$\sum_k y_{1k} + \sum_k y_{2k} = 1, \qquad y_{1k}, y_{2k} \geq 0$$

$$Q_4 = \sum_k Q_{4k}y_{1k} + \sum_k Q_{4k}y_{2k}$$

$$HF_4 = \sum_k HF_{4min}y_{1k} + \sum_k HF_{4max}y_{2k}$$

$$Q_4HF_4 = \sum_k HF_{4min}Q_{4k}y_{1k} + \sum_k HF_{4max}Q_{4k}y_{2k}$$

$$\sum_k z_{1k} + \sum_k z_{2k} = 1, \qquad z_{1k}, z_{2k} \geq 0$$

$$Q_{12} = \sum_k Q_{12k}z_{1k} + \sum_k Q_{12k}z_{2k}$$

$$HF_{12} = \sum_k HF_{12min}z_{1k} + \sum_k HF_{12max}z_{2k}$$

$$Q_{12}HF_{12} = \sum_k HF_{12min}Q_{12k}z_{1k} + \sum_k HF_{12max}Q_{12k}z_{2k}$$

Therefore, the problem can be linearized as follows:

$$\begin{aligned} \text{Minimize} \quad & \sum_k HF_{3min}Q_{3k}x_{1k} + \sum_k HF_{3max}Q_{3k}x_{2k} \\ & \sum_k HF_{4min}Q_{4k}y_{1k} + \sum_k HF_{4max}Q_{4k}y_{2k} \\ & \sum_k HF_{12min}Q_{12k}z_{1k} + \sum_k HF_{12max}Q_{12k}z_{2k} \end{aligned}$$

subject to

$$\sum_k Q_{3k}x_{1k} + \sum_k Q_{3k}x_{2k} + \sum_k Q_{4k}y_{1k} + \sum_k Q_{4k}y_{2k} = q_1$$

$$\sum_k Q_{4k}y_{1k} + \sum_k Q_{4k}y_{2k} - \sum_k Q_{7k}u_k - \sum_k Q_{11k}w_k + \sum_k Q_{12k}z_{1k} + \sum_k Q_{12k}z_{2k} = q_1$$

$$-\sum_k Q_{8k}v_k - \sum_k Q_{11k}w_k + \sum_k Q_{12k}z_{1k} + \sum_k Q_{12k}z_{2k} = q_1$$

$$R_3\sum_k Q_{3k}^2x_{1k} + R_3\sum_k Q_{3k}^2x_{2k} - R_7\sum_k Q_{7k}^2u_k - R_8\sum_k Q_{8k}^2v_k$$
$$+ \sum_k \{2[R_2q_1 + R_5(q_1+q_6)]Q_{12k} - (R_2+R_5)Q_{12k}^2\}z_{1k}$$
$$+ \sum_k \{2[R_2q_1 + R_5(q_1+q_6)]Q_{12k} - (R_2+R_5)Q_{12k}^2\}z_{2k}$$
$$-HR_8 - HF_3 = (R_2-R_1)q_1^2 + R_5(q_1+q_6)^2$$

$$R_4\sum_k Q_{4k}^2y_{1k} + R_4\sum_k Q_{4k}^2y_{2k} - HF_4 - R_8\sum_k Q_{8k}^2v_k - HR_8$$
$$+ \sum_k \{2[R_2q_1 + R_5(q_1+q_6)]Q_{12k} - (R_2+R_5)Q_{12k}^2\}z_{1k}$$
$$+ \sum_k \{2[R_2q_1 + R_5(q_1+q_6)]Q_{12k} - (R_2+R_5)Q_{12k}^2\}z_{2k}$$
$$= (R_2-R_1)q_1^2 + R_5(q_1+q_6)^2$$

$$R_9\sum_k (2q_6Q_{11k} + Q_{11k}^2)w_k - R_5\sum_k [2(q_1+q_6)Q_{12k} - Q_{12k}^2]z_{1k}$$
$$- R_5\sum_k [2(q_1+q_6)Q_{12k} - Q_{12k}^2]z_{2k} - HR_6 - HR_9 = R_5(q_1+q_6)^2 + (R_6+R_9)q_6^2$$

$$-R_7\sum_k Q_{7k}^2u_k - \sum_k[2R_{10}q_1Q_{8k} + (R_8+R_{10})Q_{8k}^2]v_k - HR_8 - HR_9$$
$$+ \sum_k[2R_9q_6Q_{11k} + (R_{11} - R_9)Q_{11k}^2]w_k = R_9q_6^2 + R_{10}q_1^2$$

$$-R_7\sum_k Q_{7k}^2u_k - \sum_k[2R_{10}q_1Q_{8k} + (R_8+R_{10})Q_{8k}^2]v_k - HR_8 + HF_{12}$$
$$+ \sum_k \{2[R_2q_1+R_5(q_1+q_6)]Q_{12k} - (R_2+R_5)Q_{12k}^2\}z_{1k} + \sum_k \{2[R_2q_1$$
$$+ R_5(q_1+q_6)]Q_{12k} - (R_2+R_5)Q_{12k}^2\}z_{2k} = (R_2+R_{10})q_1^2 + R_5(q_1+q_6)^2$$

$$\sum_k u_k = 1$$

$$\sum_k v_k = 1$$

$$\sum_k w_k = 1$$

$$\sum_k x_{1k} + \sum_k x_{2k} = 1$$

$$\sum_k y_{1k} + \sum_k y_{2k} = 1$$

$$\sum_k z_{1k} + \sum_k z_{2k} = 1$$

$$\sum_k HF_{3min}x_{1k} + \sum_k HF_{3max}x_{2k} - HF_3 = 0$$

$$\sum_k HF_{4min}y_{1k} + \sum_k HF_{4max}y_{2k} - HF_4 = 0$$

$$\sum_k HF_{12min}z_{1k} + \sum_k HF_{12max}z_{2k} - HF_{12} = 0$$

$$u_k,\ v_k,\ w_k,\ x_{1k},\ x_{2k},\ y_{1k},\ y_{2k},\ z_{1k},\ z_{2k} \geq 0$$

$$HR_6,\ HR_8,\ HR_9 \geq 0$$

The algorithm output is given in Table VI. In the same way, the algorithm can find optimal solution for other two alternatives. And the results are given in Table VII and Table VIII.

6.5 Comparison of Partial Solutions

Table IX summarizes the power consumption for each possible combination. As seen, we should put fans in Branches 10 and 12 if we consider locating two fans in the system. Similarly, Locating fans in Branches 3, 4, and 12 is the best choice

Table VI. Algorithm Output For Partial Solution of Fans in Branches 3, 4, and 12

Branch #	Q_j (m^3/s)	HF_j (Pa.)	HR_j (Pa.)
1	50.00		
2	80.50		
3	34.14	894	
4	15.87	922	
5	30.50		
6	50.00		
7	71.88		
8	56.01		
9	25.51		
10	106.01		
11	24.49		
12	130.50	1122	

Table VII. Algorithm Output For Partial Solution of Fans in Branches 3, 10, and 12

Branch #	Q_j (m^3/s)	HF_j (Pa.)	HR_j (Pa.)
1	50.00		
2	79.67		
3	38.70	168	
4	11.30		
5	29.67		
6	50.00		
7	60.97		
8	49.67		902
9	20.00		
10	99.67	522	
11	30.00		
12	129.67	1382	

Table VIII. Algorithm Output For Partial Solution of Fans in Branches 4, 10, and 12

Branch #	Q_j (m^3/s)	HF_j (Pa.)	HR_j (Pa.)
1	50.00		
2	79.82		
3	25.80		
4	24.20	323	
5	29.82		
6	50.00		
7	75.17		
8	50.97		778
9	21.15		
10	100.97	547	
11	28.85		
12	129.82	1324	

Table IX. Comparison of Power Consumption of Partial Solutions

Fan Location	Power consumption (Watt)
12	247944.46
3, 12	244179.20
4, 12	242327.00
10, 12	239507.25
3, 10, 12	237825.30
4, 10, 12	234860.50
3, 4, 12	191594.77

among the possible combinations of three fans in the system. Then, the algorithm can compare the total costs of ventilation if the following cost factors are chosen:

Annual power cost: \$450/HP

Annual maintenance cost: \$50/HP

Annualized installation and ownership cost: surface \$3,000 per fan,

Annualized installation and ownership cost: Underground \$5,000 per fan

The total annual cost of ventilation for Partial Solution (1) is:

$(450+50)\times(1926.74\times128.686)/745 + 3000 = \$169405.68$

The total annual cost of ventilation for Partial Solution (2) is:

$(450+50)\times(1389.99\times129.645 + 596\times99.5) + 3000 + 5000 = \$168743.36$

The total annual cost of ventilation for Partial Solution (3) is:

$(450+50)\times(130.399\times1122.227 + 34.135\times893.823 + 15.865\times922.447)/745$

$+ 3000 + 2\times5000 = \$141587$

These comparison are summarized in Table X.

Table X. Comparison of Total Costs of Partial Solutions

Solutions	Power Consum. (HP)	Power Cost ($)	Instal. cost ($)	Total cost ($)
1	332.811	166406	3000	169406
2	321.486	160743	8000	168743
3	257.174	128587	13000	141587

CHAPTER VII CONCLUSIONS AND RECOMMENDATIONS

7.1 Conclusions

The research presented in this dissertation solves the optimization problem of location and size of ventilation control devices. The comparison of optimization techniques of ventilation networks identified the limitations and weaknesses of the existing methods. It has also indicated the direction of this current research.

Optimization techniques for determining the location and size of control devices in mine ventilation are applied to two kinds of systems: controlled–splitting networks and semi–controlled–splitting networks. For the controlled–splitting networks, several sophisticated methods are available. However, no universally applicable solution methodology exists for the semi–controlled–splitting network. Theoretically, solving the network problem for semi–controlled splitting is more difficult than solving the network problem for controlled splitting. The semi–controlled splitting problems necessitate solution of a nonlinear, nonconvex programming model in order to determine the optimal location and size for ventilation control devices and flow distributions simultaneously.

The mathematical model attempted to capture the key features of the real problem as much as possible. The objective of the model is to minimize the cost of mine ventilation rather than minimizing power consumption. The power constraints prevent a very small fan from being located in the system. Also, the model can deal with situation when leakage exists.

The mathematical model is a nonlinear, nonconvex programming problem. The techniques of special ordered sets of variables and linear transformation have been applied to linearize the nonlinear, nonconvex model. Then the modified branch and bound procedure was used to attain the optimal solution. The algorithm can determine optimal location and size for ventilation control devices and can distribute air flow in the system to meet the legal, technical, and production requirements of airflow. It guarantees the optimal solution on the basis of computational theory. It can be used for the generalized optimization problem of ventilation networks as well as other special cases such as controlled splitting and natural splitting networks. The computerized model can serve as a tool for the design and analysis of ventilation systems.

7.2 Recommendations

Further studies are needed to extend the approaches developed in this dissertation, in the following areas:

The optimal solution obtained by using the methodology developed in this dissertation corresponds to the ventilation requirements at one point in the development of a mine. The solution may change in consideration of the requirements of ventilation for the whole life of the mine. Further research work is therefore needed in order to accommodate the dynamic nature of a mine ventilation system and determine the optimal solution with regard to long–term ventilation requirements.

The integration of computer graphics capabilities with the optimization model needs to be considered. Interactive graphics could be used to define the initial network, choose the locations of ventilation control devices, modify input data, and present the optimal solution. They would enhance the practical usefulness and ultimate value of the solution.

The proposed methodology has some restrictions on treating real mine ventilation systems, such as incompressibility of air flow and predefined topology. These limitations could be overcome through future research. The design of a compressed air flow system could be included in the optimization model, and the integration of underground mine design would allow optimization of the ventilation network topology as well as location and size of ventilation control devices.

The optimization model developed can be extended for post–optimality analysis. For example, the system parameters may change, and the air quantity requirements for the working area considering production advance, resistance factors due to extension of airway, and the lower or/and upper bounds of flow, may change from time to time. The post–optimality analysis will determine whether the optimal locations and sizes of control devices should change when these parameters vary, and the extent of parameter change within which the solution will still be optimal.

In this study the model uses a single objective, i.e. minimization of total costs of ventilation. An interesting extension would incorporate other objectives into the problem. For example, another objective could be minimization of leakage since leakage will increase power consumption. Another objective could be minimization of variations of flow quantities in branches. These objectives may contradict one another and present a multiobjective programming model. The algorithm must determine all efficient solutions.

REFERENCES

Barnes, R.J., 1983, "Optimal Mine Ventilation Design," Ph.D dissertation.

Bazaraa, M.S. and C.M.Shetty, 1979, Nonlinear Programming Theory and Algorithms, John Wiley & Sons, New York.

Beale, E.M.L, 1981, Introduction to Optimization, John Wiley & Sons.

Beale, E.M.L. and J.A.Tomlin, 1970, "Special Facilities in a General Mathematical Programing System for Nonconvex Problems Using Ordered Sets of Variables," Proc. of the Fifth International Conference on Operational Research, Ed. J. Lawrence, pp. 447–454.

Beale, E.M.L. and J.J.H.Forrest, 1976, "Global Optimization Using Special Ordered Sets," Mathematical Programming 10 (1976), pp. 52–69.

Beale, E.M.L. and J.J.H.Forrest, 1978, "Global Optimization as an Extension of Inter Programming," Towards Global Optimization 2, Ed. L.C.W. Dixon and G.P. Szago, North–Holland, Amsterdam.

Beale, E.M.L., 1980, "Branch and Bound Methods for Numerical Optimization of Non–convex functions," COMPSTAT 80 Proceedings in Computational Statistics, Ed. M.M.Barritt and D. Wishart, North–Holland, Amsterdam, pp. 221–269.

Beale, E.M.L., 1985, "The Evolution of Mathematical Programming Systems," Journal of Operational Research, Val.36, No.5, pp. 357–366.

Beale, E.M.L., 1979, "Branch and Bound Methods for Mathematical Programing Systems," Annals of Discrete Mathematics 5, pp. 201–219.

Beale, E.M.L., 1975, "Some Uses of Mathematical Programming Systems to Solve Problems that are not Linear," Operational Research Quarterly, Pergamon Press, Vol.26, 3, ii, pp. 609–618.

Bhamidipati, S. and E. Topuz, 1983, "A Critical Path Crashing Technique to Optimize Multiple Fan Ventilation systems," Preprint, SME–AIME Annual,meeting, Atlanta, Georgia, March.

Calizaya,F., M.J. Mcpherson, and P. Mousse–Jones, 1986, "An Algorithm for Selecting the Optimum Combination of Main and Booster Fans in Underground Mines," Proc. of the Third U.S. Mine Ventilation Symposium, the Pennsylvania State University, University Park, PA, October, pp. 408–417.

Faiz A Al–khayyal and James E Falk, 1983, " Jointly Constrained Biconvex Programming," Mathematics of Operations Research 8, May, pp. 273–286.

Forrest,H.J.J., J.P.H.Hirst and J.A.Tomlin, 1974, "Practical Solution of Large and Complex Integer Programming Problems with UMPIRE," Management Science 20, pp. 736–773.

Greenwood, A. G., 1984, "A Decision Support system for Tuition and Fee Policy Analysis," Ph.D. dissertation, College of Business, Virginia Polytechnic Institute and State University, Blacksburg, Virginia.

Hillier, F.S., and G.J.Lieberman, 1986, **Introduction to Operations Research**, Holden–day, Inc., Oakland, California.

Jones, M., M.J. Mcpherson, and P. Mousset–Jones, 1986, " A study of Booster Fan in the Simulation of Mine Ventilation Networks," Proc. of the Fourth Workshop of Generic Mineral Technology Center, Mine System Design and Ground Control, Moscow, Idaho, October, pp. 105–114.

Johnson, T.B., R.J.Barnes, and P.H.King, 1982, "The Optimal Controlled Flow Mine Ventilation Problem: An Operations Research Approach," **Preprint**, for presentation at the 1st International SME–AIME Fall Meeting, Honolulu, Hawaii, September.

Kalasky, D.R. and J.M. Mutmansky, 1986, "Development and Application of a Multi–period, Time–dependent Ventilation Network Analyzer," Proc. of the 19th APCOM Symposium, England, pp. 587–596.

Lasdon, L.S., 1970, **Optimization Theory for Large Systems**, Macmillan, New York.

McPherson, M.J, N. Dhar and P. Mousset–Jones, 1985, "A Survey of Booster Fan Use in Underground Mines," Proc. of The 3rd Workshop of Generic Mineral

Technology Center, Mine Systems Design and Ground Control, Virginia Polytechnic Institute & State University, Blacksburg, VA, pp.173–191.

Miller, C.E., 1963, "The Simplex Method for Local Separable Programming," **Recent Advances in Mathematical Programming**, Ed. R.L.Graves and P.Wolfe, McGraw Hill, New York, pp. 89–100.

Mitra Gautam, 1976, **Theory and Application of Mathematical Programming**, Academic Press, New York.

Schrage L., 1981, "User's Manual for LINDO", Palo Alto, California, The Scientific Press.

Topuz, E., S.S.Bhamidipati, M. Bartkoski, 1982, "Improvement of Ventilation system Efficiency Through the Analysis of Air Leakage," Proc. of the 1st US Mine Ventilation Symposium, University of Alabama, Tuscaloosa, AL, pp.285–289.

Tomlin, J.A., 1970,"Branch and Bound Methods for Integer and Nonconvex Programming, **Integer and Nonlinear Programming**, North Holland, Amsterdam, pp.437–450.

Wang Z.C. and E.Y.Yao, 1984, "Optimum Method of Regulating a Ventilation Network," Proc. of the 3rd International Mine Ventilation Congress, Harrogate, England, pp. 53–58.

Wang, Y.J. and M.T.Pana, 1971, "Solving Mine Ventilation Network Problems by Linear Programming:," Preprint, AIME Annual Meeting, New York,

Wang, Y.J, 1981, "A Critical Path Approach to Mine ventilation Networks With Controlled Flow," SME Preprint No 81–83, SME/AIME Annual Meeting, Chicago, Illinois.

Wang, Y.J., 1982, "Ventilation Network Theory," Mine Ventilation and Air Conditioning, H.L. Hartman, et al.(ed), New York, John Wiley and sons, pp.483–516.

Wang, Y.J., 1983, "Characteristics of Multiple–fan Ventilation Networks," SEM Preprint No. 83–144, SME/AIME Annual Meeting, Atlanta, Georgia.

Wang, Y.J., 1989, "Procedure for Solving a More Generalized System of Mine Ventilation Network Equations," Proc. of the 4th US Mine Ventilation Symposium, University of California, Berkeley, California, pp.419–424.

Wu Xing and E. Topuz, 1987, "The Determination of Booster Fan Locations in Underground Mines," Proc. of the Third U.S. Mine Ventilation Symposium, The Pennsylvania State University, University Park, Pa, October, pp.401–407.

Wu Xing and E. Topuz, 1989, "Comparison of Methods for Determination of Booster Fan Locations in Underground Mines," Proc. of the 4th US Mine Ventilation Symposium, University of California, Berkeley, California, pp. 355–362.

Wu Xing, E. Topuz, and M. Karfakis, 1991, "Optimization of Ventilation Control Device Locations and Sizes in Underground Mine Ventilation Systems," Proc. of the 5th US Mine Ventilation Symposium, West Virginia University, Morgantown, West Virginia, pp. 391–399.

APPENDICES

APPENDIX A REDUCED COSTS

Consider the problem

Minimize $c_1x_1 + c_2x_2 + \ldots + c_Nx_N$

subject to

$$\begin{array}{l} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1N}x_N = b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2N}x_N = b_2 \\ \vdots \qquad \vdots \qquad\qquad \vdots \qquad \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mN}x_N = b_m \end{array} \tag{A.1}$$

where $N = m + n$. The problem may be transformed by carrying out m steps of Gaussian elimination with back substitution: at end of these steps the reduced matrix should contain an $m \times m$ unit matrix and the corresponding dependent variables are expressed in terms of n remaining variables whose column entries make up the rest $(m \times n)$ of the matrix, the r.h.s. values are also simultaneously transformed. At the end of these steps the system of equations reduces to a canonical form.

If we include the objective function with the rest of the equations, then a new variable x_0 is introduced and the objective form is rewritten as

$$x_0 + \sum_{j=1}^{N} c_jx_j = b_0$$

The system of equation in (A.1) can now be expressed in terms of an (m + 1) × (N + 1) matrix

$$AX = b \tag{A.2}$$

Let the matrix A be partitioned such that A = (N | B), (A.2) can be then expressed as

$$(N \mid B)\begin{bmatrix} X_N \\ X_B \end{bmatrix} = b \tag{A.3}$$

where $X_N = (x_1, x_2, \ldots, x_n)^T$, and $X_B = (x_0, x_{n+1}, \ldots, x_{n+m})^T$.

Writing out in full this takes the form

$$\begin{array}{l|l}
c_1x_1 + c_2x_2 + \ldots + c_nx_n + & x_0 + c_{n+1}x_{n+1} + \ldots + c_Nx_N = b_0 \\
a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n + & 0 + a_{1n+1}x_{n+1} + \ldots + a_{1N}x_N = b_1 \quad (A.4) \\
a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n + & 0 + a_{2n+1}x_{n+1} + \ldots + a_{2N}x_N = b_2 \\
\vdots \qquad \vdots \qquad N & \qquad B \quad \vdots \; \vdots \qquad \vdots \qquad \vdots \\
a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n + & 0 + a_{mn+1}x_{n+1} + \ldots + a_{mN}x_N = b_m
\end{array}$$

Note that B is an (m + 1) × (m + 1) square submatrix and N represents the rest of the partitioned columns of A. If B is not singular, i.e. determinant of B is not zero and B^{-1} exists, then the equation system can be solved for only

$$BX_B = b, \tag{A.5}$$

and B is said to constitute a basis matrix of the equality system (A.2), (A.3), (A.4). Premultiplying (A.3) and (A.4) by B^{-1}, it follows that

$$(B^{-1}N \mid B^{-1}B)X = B^{-1}b = \beta$$

or
$$(B^{-1}N \mid I)X = B^{-1}b = \beta \qquad \text{(A.6)}$$

If $\bar{a}_{ij}$ denotes a transformed element of the matrix then (A.6) may be expressed as

$$\begin{array}{lllll}
\bar{a}_{01}x_1 + \bar{a}_{02}x_2 + \dots + \bar{a}_{0n}x_n + x_0 & & & & = \beta_0 \\
\bar{a}_{11}x_1 + \bar{a}_{12}x_2 + \dots + \bar{a}_{1n}x_n + & x_{n+1} & & & = \beta_1 \\
\bar{a}_{21}x_1 + \bar{a}_{22}x_2 + \dots + \bar{a}_{2n}x_n + & & x_{n+2} & & = \beta_2 \qquad \text{(A.7)} \\
\quad \vdots \qquad \vdots & & & & \\
\bar{a}_{m1}x_1 + \bar{a}_{m2}x_2 + \dots + \bar{a}_{mn}x_n + & & & x_{n+m} & = \beta_m
\end{array}$$

If the inverse matrix is at hand, a matrix product $B^{-1}a_j$ where a_j is a column of the original matrix produces the updated column $\bar{a}_j = B^{-1}a_j$, $\bar{a}_j$ being the transformed column (of index j) of the transformed matrix/tableau.

If the basis matrix B corresponds to variable x_0 and the last m variables x_{n+1}, $x_{n+2}, \dots, x_{n+m}$. The matrix B and its inverse are set out below

$$B = \begin{bmatrix} 1 & c_{n+1} & \cdots & c_{n+m} \\ 0 & a_{1n+1} & \cdots & a_{1n+m} \\ \vdots & \vdots & & \vdots \\ 0 & a_{mn+1} & & a_{mn+m} \end{bmatrix}$$

$$B^{-1} = \begin{bmatrix} 1 & \pi_1 & & \pi_m \\ 0 & \bar{a}_{1n+1} & \cdots & \bar{a}_{1n+m} \\ \vdots & \vdots & & \vdots \\ 0 & \bar{a}_{mn+1} & & \bar{a}_{mn+m} \end{bmatrix} = \begin{bmatrix} \pi_0 & \pi_1 & & \pi_m \\ 0 & \bar{a}_{1n+1} & \cdots & \bar{a}_{1n+m} \\ \vdots & \vdots & & \vdots \\ 0 & \bar{a}_{mn+1} & & \bar{a}_{mn+m} \end{bmatrix} \qquad \text{(A.8)}$$

Note that this B^{-1} matrix must contain the unit column corresponding to the variable x_0 and always occupying the basis. The first component of the transformed column vector $\bar{a}_j$ may therefore be expressed as

$$\begin{aligned} \bar{c}_j = \bar{a}_{0j} &= (\text{first row of } B^{-1})(\text{column } a_j) \\ &= (\pi_0, \pi_1, \pi_2, \ldots, \pi_m)(c_j, a_{1j}, a_{2j}, \ldots, a_{mj})^T \qquad \text{(A.9)} \\ &= (1, \pi_1, \pi_2, \ldots, \pi_m)(c_j, a_{1j}, a_{2j}, \ldots, a_{mj})^T \\ &= c_j + \sum_{i=1}^{m} a_{ij}\pi_i = c_j + z_j \\ &= \sum_{i=0}^{m} a_{ij}\pi_i \end{aligned}$$

where $\bar{\pi} = (\pi_1, \pi_2, \ldots, \pi_m) = -C\bar{B}^{-1}$ in which

$$C = (c_{n+1}, c_{n+2}, \ldots, c_{n+m})$$

$$\bar{B} = \begin{bmatrix} a_{1n+1} & a_{1n+m} \\ \vdots & \vdots \\ a_{mn+1} & a_{mn+m} \end{bmatrix}$$

It can be proved by the following example, Let

$$M = \begin{bmatrix} I & Q \\ 0 & R \end{bmatrix}$$

where 0 denotes a matrix with all zero entries. Then

$$M^{-1} = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$$

satisfies

$$MM^{-1} = \begin{bmatrix} I & Q \\ 0 & R \end{bmatrix} \begin{bmatrix} A & B \\ C & D \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix}$$

which implies the following matrix equation gives

$$A + QC = I, \quad B + QD = 0, \quad RC = 0, \quad RD = I.$$

Solving these simultaneous equations gives

$$C = 0, \quad A = I, \quad D = R^{-1}, \text{ and } \quad B = -QR^{-1}$$

or, equivalently,

$$M^{-1} = \begin{bmatrix} I & -QR^{-1} \\ 0 & R^{-1} \end{bmatrix}$$

For the special ordered sets variables of the linearized problem in Eqs. (4.13),

$$c_j = g_{0j}(x_j), \qquad a_{ij} = g_{ij}(x_j)$$

Thus

$$\begin{aligned}
\bar{c}_j = \bar{a}_{0j} &= (\text{first row of } B^{-1})(\text{column } a_j) \\
&= (\pi_0, \pi_1, \pi_2, \ldots, \pi_m)(g_{0j}, g_{1j}, g_{2j}, \ldots, g_{mj})^T \\
&= (1, \pi_1, \pi_2, \ldots, \pi_m)(g_{0j}, g_{1j}, g_{2j}, \ldots, g_{mj})^T \\
&= g_{0j} + \sum_{i=1}^{m} \pi_i g_{ij}(x_j) \\
&= \sum_{i=0}^{m} \pi_i g_{ij}(x_j) \qquad \text{(A.10)}
\end{aligned}$$

This is exactly expression for reduced cost in Section 4.5.2.

From (A.7), $x_0 = \beta_0 - (\bar{a}_{01}x_1 + \bar{a}_{02}x_2 + \ldots + \bar{a}_{0n}x_n)$. So, if all the coefficients $\bar{a}_{0j}$ (the reduced costs) are positive the trial solution is optimum. Otherwise we select a negative one, say $\bar{a}_{0q}$. Theoretically it does not matter which negative reduced cost we take, but in practice we usually take the most negative.

When we use the special ordered sets variables, we add convexity row $\sum_k \lambda_{jk} = 1$ to the constraint set. As a result, the reduced cost for the variable λ_j is

$$\bar{c}_j = g_{0j} - \pi \begin{bmatrix} g_{ij} \\ \hline 1 \end{bmatrix} \qquad \text{(A.11)}$$

Partition π as $\qquad \pi = (\bar{\pi}_1, \ \pi_0)$

where π_1 corresponds to π in (A.10), π_0 to the single constraint $\sum_k \lambda_{jk} = 1$.

Therefore, (A.11) becomes;

$$\bar{c}_j = g_{0j} - \bar{\pi}_1 g_{ij} - \pi_0 = (g_{0j} - \bar{\pi}_1 g_{ij}) - \pi_0$$

where $-\bar{\pi}_1 = (\pi_1, \pi_2, \ldots, \pi_m)$ by the definition above. Hence,

$$\bar{c}_j = \sum_i \pi_i g_{ij} - \pi_0$$

According to the criterion above, we choose a variable λ_j with the most reduced cost to enter the basis in order to improve the solution, that is, solve the following problem

$$\text{Minimize } z_j = \sum_i \pi_i g_{ij}(x_j) - \pi_0 \qquad \text{(A.12)}$$

subject to

$$L_j \leq x_j \leq U_j$$

where π_0 is scalar, then equivalently we solve the problem below

$$\text{Minimize } z_j = \sum_i \pi_i g_{ij}(x_j)$$

subject to

$$L_j \leq x_j \leq U_j$$

If $z_j \geq \pi_0$, the trail solution is optimum, no λ_j can enter the basis; Otherwise we introduce the λ_j into the basis.

After using the differences of squares or logarithm method to convert the product terms in the objective function of the ventilation optimization problem, we can rewrite the problem here without losing generality as

$$\text{Minimize } z = \sum_j \sum_k f_j(x_{jk})\lambda_{jk} \tag{A.13}$$

subject to
$$\sum_j \sum_k g_{ij}(x_{jk})\lambda_{jk} = b_0 \tag{A.14}$$

$$\sum_k \lambda_{jk} = 1 \tag{A.15}$$

$$\lambda_{jk} \geq 0 \tag{A.16}$$

Multiply (A.14) by shadow prices $\bar{\pi}_1$ and (A.15) by π_{0j}, sum and subtract from the cost equation, yielding

$$z - \bar{\pi}_1 b_0 - \sum_j \pi_{0j} = \sum_j \sum_k \lambda_{jk}(f_j - \bar{\pi}_1 g_{ij} - \pi_{0j}) \tag{A.17}$$

The quantity in parentheses on the right of (A.17) is simply the reduced cost for λ_{jk}. By solving the subproblem (A.12), we compute min z_j for each j. Since λ_{jk} is nonnegative, z_j may be replaced by its minimum value and the right–hand side of (A.17) cannot increase, so

$$z - \bar{\pi}_1 b_0 - \sum_j \pi_{0j} \geq \sum_j \sum_k \lambda_{jk}(\min z_j) \tag{A.18}$$

$$z - \bar{\pi}_1 b_0 - \sum_j \pi_{0j} \geq \sum_j (\min z_j) \sum_k \lambda_{jk} \tag{A.19}$$

Using (A.15), (A.19) becomes

$$z \geq \sum_j (\min z_j) + \bar{\pi}_1 b_0 + \sum_j \pi_{0j} \tag{A.20}$$

Since (A.20) holds for all values of z obtainable from (A.13) to (A.16), it holds for the minimum values, so

$$\min z \geq \sum_j (\min z_j) + \bar{\pi}_1 b_0 + \pi_0 e_p \tag{A.21}$$

where e_p is the n–dimensional sum vector. Relation (A.21) may be placed in a more compact form by writing the last two terms on the right as

$$(\bar{\pi}_1, \pi_0)\binom{b_0}{e_p} = c_B B^{-1}\binom{b_0}{e_p} = c_B x_B = z_B \tag{A.22}$$

where z_B is the value of z associated with the current basis, B. Thus (A.21) becomes

$$\min z \geq z_B + \sum_j (\min z_j) \tag{A.23}$$

which is the desired lower bound of the problem in Eqs.(A.13) to (A.16). If we replace z by $-x_0$ then (A.23) becomes

$$\max x_0 \leq z_B - \sum_j (\min z_j) \tag{A.24}$$

This is the upper bound for the problem in Section 5.1.

APPENDIX B THEOREM AND PROOF

THEOREM

Consider the Problem P to minimize $\sum_{j=1}^{n} f_j(x_j)$ subject to $\sum_{j=1}^{n} g_{ij}(x_j) \leq b_i$ for $i = 1,\ldots,m$, and $x_j \geq 0$ for $j = 1,\ldots,n$. Let $L = \{j\text{: } f_j \text{ and } g_{ij} \text{ are linear}\}$. Suppose that for $j \notin L$, f_j is strictly convex and that g_{ij} is convex for $i = 1,\ldots,m$. Suppose that for each $j \notin L$, f_j and g_{ij} for $i = 1,\ldots,m$ are replaced by their piecewise linear approximations via the grid points x_{jk} for $k = 0,\ldots,K_j$, this yields the equivalent linear program defined below:

$$\text{Minimize } \sum_{j \in L} f_j(x_j) + \sum_{j \notin L} \sum_{k} \lambda_{jk} f_j(x_{jk})$$

subject to

$$\sum_{j \in L} g_{ij}(x_j) + \sum_{j \notin L} \sum_{k} \lambda_{jk} g_{ij}(x_{jk}) \leq b_i \quad \text{for } i = 1,\ldots,m \qquad \text{(B.1)}$$

$$\sum_{k} \lambda_{jk} = 1 \qquad \text{for } j \notin L$$

$$\lambda_{jk} \geq 0 \qquad \text{for } k = 0,\ldots,K_j, \text{ and } j \notin L$$

$$x_j \geq 0 \qquad \text{for } j \in L$$

Let $\bar{x}_j$ for $j \in L$ and $\bar{\lambda}_{jk}$ for $k = 0,\ldots,K_j$ and $j \notin L$ solve the above problem. Then:

For each $j \notin L$, at most two $\overline{\lambda}_{jk}$ are positive, and they must be adjacent.

Proof

To prove this, it suffices to show that for each $j \notin L$, if $\overline{\lambda}_{jl}$ and $\overline{\lambda}_{jp}$ are positive, then the grid points x_{jl} and x_{jp} must be adjacent. By contradiction, suppose that there exist $\overline{\lambda}_{jl}$ and $\overline{\lambda}_{jp} > 0$, where $\overline{x}_{jl}$ and $\overline{x}_{jp}$ are not adjacent. Then, there exist a grid point $x_{jo} \in (x_{jl}, x_{jp})$ that can be expressed as $x_{jo} = \alpha_1 x_{jl} + \alpha_2 x_{jp}$, where α_1, $\alpha_2 > 0$ and $\alpha_1 + \alpha_2 = 1$. Now consider the optimal solution to the problem defined by (B1). Let $u_i \geq 0$ for $i = 1,\dots,m$ be the optimum Lagrangian multipliers associated with the first m constraints, and for each $j \notin L$, let v_j be the optimal Lagrangian multiplier associated with the constraint $\sum_k \lambda_{jk} = 1$. Then the following subset of the Kuhn–Tucker necessary conditions are satisfied:

$$f_j(x_{jl}) + \sum_{i=1}^{m} u_i g_{ij}(x_{jl}) + v_j = 0 \tag{B.2}$$

$$f_j(x_{jp}) + \sum_{i=1}^{m} u_i g_{ij}(x_{jp}) + v_j = 0 \tag{B.3}$$

$$f_j(x_{jk}) + \sum_{i=1}^{m} u_i g_{ij}(x_{jk}) + v_j \geq 0 \quad \text{for } k = 1,\dots,K_j \tag{B.4}$$

By strict convexity of f_j and convexity of g_{ij} and by (B.2) and (B.3), we have

$$f_j(x_{jo}) + \sum_{i=1}^{m} u_i g_{ij}(x_{jo}) + v_j < \alpha_1 f_j(x_{jl}) + \alpha_2 f_j(x_{jp})$$

$$+ \sum_{i=1}^{m} u_i[\alpha_1 g_{ij}(x_{j1}) + \alpha_2 g_{ij}(x_{jp})] + v_j = 0$$

This contradicts (B.4) for $k = o$, and hence x_{j1} and x_{jp} must be adjacent, and this complete the proof.

APPENDIX C ESTIMATED DEGRADATION

This approach was written for integer variables, but the same logic can be used for special ordered sets. Corresponding to each unsatisfied integer variable x_j we find some way of computing D_j^-, the estimated reduction in v_0, the value x_0 at the optimal solution to a linear programming subproblem, from reducing U_j to n_j, and D_j^+, the estimated reduction in v_0 from increasing L_j to $n_j + 1$, where $L_j \leq x_j \leq U_j$, and $v_j = n_j + f_j$, n_j is an integer and $0 \leq f_j < 1$, v_j is the value of the integer variable x_j at the optimal solution to a linear programming subproblem. D_j^+ and D_j^- are known as estimated degradations form driving x_j down and up respectively.

Consider the effect of imposing a change on the value of some integer variable x_k. It is natural to rewrite the constraints with the variable on the right hand side. The problem then becomes:

Maximize x_0

subject to

$$x_0 + \sum_{j=k} a_{0j}x_j = b_0 - a_{0k}x_k$$

$$\sum_{j=k} a_{ij}x_j = b_i - a_{ik}x_k \qquad i = 1,\ldots, m$$

$$L_j \leq x_j \leq U_j$$

This shows that if we increase the trial value of x_k by $1 - f_k$, then there is no effect on the value of x_0 or any other variables if we simultaneously decrease each b_i by $z_i = a_{ik}(1 - f_k)$. The same argument applies to decreasing x_k by f_k if we put $z_i = -a_{ik}f_k$. So, to evaluate the effect of changing x_j, we imagine that the value of this variable is held constant while we decrease each b_i by z_i, operating simultaneously on all rows. If π_i denotes the shadow price on ith row, then if all z_i were small, v_0 would be degraded $\sum_i \pi_i z_i$. It is guaranteed lower bound. We can find an upper bound on the degradation in terms of the minimum and maximum shadow prices $\pi_{\min i}$ and $\pi_{\max i}$. If P denotes the set of rows for which $z_i > 0$ and N the set of rows for which $z_i < 0$, then

$$\sum_i \pi_i z_i \leq D \leq \sum_{i \in P} \pi_{\max i} z_i + \sum_{i \in N} \pi_{\min i} z_i$$

So, we can write

$$D = \sum_i \pi_i z_i + \sum_i r_i$$

where

$$0 \leq r_i \leq (\pi_{\max i} - \pi_i) z_i \qquad \text{for } i \in P$$

$$0 \leq r_i \leq (\pi_{\min i} - \pi_i) z_i \qquad \text{for } i \in N$$

But these upper bound are not necessarily useful and may even be infinite. It seems that only heuristic methods can provide realistic estimates for r_i. If the reductions on the right hand sides are written as θz_i, then we may consider how the shadow prices π_i vary as θ increases from 0 to 1. Elementary theory shows that $\sum_i \pi_i z_i$ increases monotonically. Although it is piecewise constant, and increases discontinuously. But the increases may stop, and the individual p_i will always lie within the bounds for the shadow prices, which may be far tighter than indicated by our preliminary analysis. So, the further adjustable parameters namely pseudo shadow prices π_{li} and π_{ui} and a small positive tolerance T_{pi} are introduced. We may estimate D as if p_i varied linearly with z_i between limits of

$$\pi_{lai} = \max(\pi_{mini}, \min(\pi_{li}, \pi_i - T_{pi}))$$

$$\pi_{uai} = \min(\pi_{maxi}, \max(\pi_{ui}, \pi_i + T_{pi}))$$

with a rate of change of π_{di}. Since $r_i = \int_{z=0}^{z_i} (p_i(z) - \pi_i)d_z$, and $p_i(0) = \pi_i$, these assumption imply that, for $i \in P$

$$r_i = \frac{1}{2}\pi_{di}z_i^2 \qquad \text{if } \pi_{di}z_i < \pi_{uai} - \pi_i$$

$$r_i = (\pi_{uai} - \pi_i)(z_i - \frac{1}{2}(\pi_{uai} - \pi_i)/\pi_{di}) \qquad \text{otherwise}$$

and for $i \in N$

$$r_i = \frac{1}{2}\pi_{di}z_i^2 \qquad \text{if } \pi_{di}z_i > \pi_{lai} - \pi_i$$

$$r_i = (\pi_{lai} - \pi_i)(z_i - \frac{1}{2}(\pi_{lai} - \pi_i)/\pi_{di}) \qquad \text{otherwise}$$

Note that, from $i \in P$, $r_i > 0$ unless $\pi_i = \pi_{maxi}$, and for $i \in N$, $r_i > 0$ unless $\pi_i =$

π_{mini}. So, the estimated degradation is never zero unless the true degradation is zero.

The optimal parameter settings for this approach are not clear. Good estimates of these quantities should be used when available, but underestimates are less disastrous than overestimates. So,we can set small positive values, P_i^+, for all π_{ui}, small negative values, P_i^-, for all π_{li} and effectively infinite values for all π_{di}. Then the estimated degradation becomes:

$$D = \sum_i \max\{P_i^+ z_i,\ P_i^- z_i,\ \pi_i z_i\}$$

For the special ordered sets, $z_i = (g_{cij} - g_{aij})$; if we set P_i^- as small positive value, then the estimated degradation, D_{A_j}, resulting from using the corrected contribution instead of the actual contribution is

$$D_{A_j} = \sum_i \max\{P_i^+(g_{cij} - g_{aij}),\ -P_i^-(g_{cij} - g_{aij}),\ \pi_i(g_{cij} - g_{aij})\}$$

or $D_{A_j} = \sum_i C_i$

where $C_i = \max(P_i^+, \pi_i)(g_{cij} - g_{aij}) \qquad \text{if } g_{cij} - g_{aij} \geq 0$

$$C_i = \max(P_i^-, -\pi_i)\,|g_{cij} - g_{aij}| \qquad \text{if } g_{cij} - g_{aij} < 0$$

For the ventilation problem mentioned, the estimated degradation for the constraints with respect to Kirchhoff's first law, for objective function, and power constraints can be calculated as follows:

For first law constraints,

$$g_{cj} = (\sum_k \lambda_{jk})g_j(\bar{Q}_j) = (\sum_k \lambda_{jk})g_j(\sum_k \lambda_{jk}Q_{jk}/\sum_k \lambda_{jk}) = (\sum_k \lambda_{jk})\sum_k \lambda_{jk}Q_{jk}/\sum_k \lambda_{jk}$$
$$= \sum_k \lambda_{jk}Q_{jk}$$

$$g_{aj} = \sum_k \lambda_{jk}g_j(Q_{jk}) = \sum_k \lambda_{jk}Q_{jk}$$

$$g_{cj} - g_{aj} = (\sum_k \lambda_{jk})g_j(\bar{Q}_j) - \sum_k \lambda_{jk}g_j(Q_{jk}) = \sum_k \lambda_{jk}Q_{jk} - \sum_k \lambda_{jk}Q_{jk} = 0$$

For the objective function or power constraints,

$$g_{cj} = (\sum_k \lambda_{jk})g_j(\bar{Q}_j) = (\sum_k \lambda_{jk})HF_j\sum_k \lambda_{jk}Q_{jk}/\sum_k \lambda_{jk} = HF_j\sum_k \lambda_{jk}Q_{jk}$$

$$g_{aj} = \sum_k \lambda_{jk}g_j(Q_{jk}) = \sum_k \lambda_{jk}Q_{jk}HF_j = HF_j\sum_k \lambda_{jk}Q_{jk}$$

$$g_{cj} - g_{aj} = (\sum_k \lambda_{jk})g_j(\bar{Q}_j) - \sum_k \lambda_{jk}g_j(Q_{jk}) = HF_j\sum_k \lambda_{jk}Q_{jk} - HF_j\sum_k \lambda_{jk}Q_{jk} = 0$$

Therefore, calculation of degradation is carried out only for functions with respect to the second law constraints.

APPENDIX D GLOBAL OPTIMIZATION IN ONE DIMENSION

Our problem is to find a value of x that gives f(x) a value within some tolerance ϵ of its global minimum when it is not necessarily convex.

To derive a finite method, we first make some additional assumptions:

(a) The function f(x) is twice differentiable,

(b) We can evaluated f and f' for any x within $L \leq x \leq U$,

(c) We can derive finite upper and lower bounds on f'' within any interval.

(d) The total interval $L \leq x \leq U$ can be divided into a finite number of subintervals such that each $g_i''(x)$ is monotonic within each subinterval.

We now have the inequalities that, for $l_j \leq x \leq u$, $M_1 \leq f'' \leq M_2$, where

$$M_1 = \sum_i \min(\pi_i g_i''(l_j),\ \pi_i g_i''(u)),$$

$$M_2 = \sum_i \max(\pi_i g_i''(l_j),\ \pi_i g_i''(u)) \qquad \text{(D.1)}$$

We start by calculating f at the end points of all subintervals over which the g_i'' are monotonic and define f_{min} as the smallest of these values of f and x_{min} as the corresponding value of x.

We now must see whether there is any possibility that $f < f_{min} - \epsilon$ within any subinterval. We therefore compute M_1 and M_2 for the subinterval and construct a parabola with second derivative defined by M_1 and with the same function value and first derivative as f at the lower end of the subinterval. This parabola defines a lower bound on the value of f(x) anywhere within the subinterval. We find another lower bound by constructing another parabola with the same function value and first derivative as f at upper end of the subinterval. We now sharpen the bounds by constructing another parabola with second derivative M_2 touching each of the first two parabolas. The piecewise parabolic function $\bar{g}(x)$ defined by these three parabolas thus defines a lower bound of f for any x within the subinterval. If the minimum of this lower bound function exceeds $f_{min} - \epsilon$, then we can exclude this subinterval and proceed to the next one. If not, we divide the subinterval into two further subintervals at the point where the lower bound function is minimized.

We note that if $M_2 \le 0$ within a subinterval, then f is concave within this subinterval and there is no possibility of an interior minimum. Equally, if $M_1 \ge 0$, then f is convex and we need only search for a local minimum within this subinterval. We now derive formulae for the minimum of the function $\bar{g}(x)$ for $0 \le x \le h$ under the conditions that

$$\bar{g}(0) = f(0), \qquad \bar{g}'(0) = f'(0)$$

$$\bar{g}(h) = f(h), \qquad \bar{g}'(h) = f'(h),$$

$$M_1 \leq \bar{g}'' \leq M_2$$

It is fairly obvious that the minimum is achieved by putting $\bar{g}'' = M_1$ for $0 < x < x_1$ and $x_2 < x < h$, and $\bar{g}'' = M_2$ for $x_1 < x < x_2$, for some x_1 and x_2. We therefore derive formulae for x_1 and x_2 based on continuity requirements. We see that

$$\bar{g}'(x_1) = f'(0) + M_1 x_1$$

and $\bar{g}(x_1) = f(0) + f'(0)x_1 + M_1 x_1^2/2$

hence $\bar{g}'(x_2) = \bar{g}'(x_1) + M_2(x_2 - x_1)$

$$= f'(0) + (M_1 - M_2)x_1 + M_2 x_2$$

and $\bar{g}(x_2) = \bar{g}(x_1) + \bar{g}'(x_1)(x_2 - x_1) + M_2(x_2 - x_1)^2/2$

which reduces to

$$\bar{g}(x_2) = f(0) + f'(0)x_2 + (M_2 - M_1)x_1^2/2 + (M_1 - M_2)x_1 x_2 + M_2 x_2^2/2$$

Hence $\bar{g}'(h) = \bar{g}'(x_2) + M_1(h - x_2)$

$$= f'(0) + (M_1 - M_2)(x_1 - x_2) + M_1h \tag{D.2}$$

and $\bar{g}(h) = \bar{g}(x_2) + \bar{g}'(x_2)(h - x_2) + M_1(h - x_2)^2/2,$

which reduces to

$$\bar{g}(h) = f(0) + f'(0)h + M_1h^2/2$$

$$+ (M_1 - M_2)(x_2 - x_1)(x_2 + x_1 - 2h)/2 \tag{D.3}$$

But $\bar{g}'(h) = f'(h)$ and $\bar{g}(h) = f(h)$, so if we write

$$a = f'(h) - f'(0) - M_1h$$

$$b = f(h) - f(0) - f'(0)h - M_1h^2/2$$

We deduce from (D.2) that

$$x_2 - x_1 = a/(M_2 - M_1) \tag{D.4}$$

and from (D.3) that

$$b = (M_1 - M_2)(x_2 - x_1)(x_2 + x_1 - 2h)/2$$

$$= a(2h - x_1 - x_2)/2$$

so that $x_1 + x_2 = 2h - 2b/a$ (D.5)

hence, from (D.4) and (D.5),

$$x_1 = h - b/a - a/2(M_2 - M_1)$$

We also write

$$c = f'(0) + M_1x_1$$

then, for $x_1 < x < x_2$,

$$\bar{g}'(x) = c + M_2(x - x_1),$$

so $\bar{g}'(x) = 0$ when $x = x_1 - c/M_2$, and at this point

$$\bar{g}(x) = f(0) + f'(0)x_1 + M_1x_1^2/2 + c(x - x_1) + M_2(x - x_1)^2/2$$

which reduces to

$$\bar{g}(x) = f(0) + f'(0)x_1/2 + c(x_1 - c/M_2)/2 \qquad \text{(D.6)}$$

The right–hand side of (D.6) defines the minimum of g(x) if

$$0 \leq -c/M_2 \leq x_2 - x_1 = a/(M_2 - M_1)$$

If $-c/M_2$ lies outside this range, then no interior minimum is possible unless $M_1 > 0$. But if $M_1 > 0$ and $c > 0$ the minimum is

$$f(0) - (f'(0))^2/M_1, \text{ taken when } x = -f'(0)/M_1,$$

while if $M_1 > 0$ and $-c/M_2 > a/(M_2 - M_1)$ the minimum is

$$f(h) - (f'(h))^2/2M_1, \text{ taken when } x = h - f'(h)/M_1$$

We can now define the algorithm for the global minimization of the function $f(x)$ of the scalar variable x within the range $L \leq x \leq U$. We assume that $f(x)$ is defined by (4.14) and that $g''_i(x)$ is monotonic with any subinterval $p_k < x < p_{k+1}$ for any $k < K$, where

$$L = p_1 < p_2 < \ldots < p_k = U$$

We also assume that we are given a positive tolerance ϵ. We start by defining $l_k = p_k$ for $k = 1, \ldots, K - 1$; and compute and store $f(x)$, $f'(x)$ and $g''_i(x)$ for $x = l_k$. We put $u = U$ and compute and store $f(u)$, $f'(u)$ and $g''_i(u)$. Put

$$f_{min} = \min[\min f(l_k), f(u)]$$

and define

$$x_{min} \text{ such that } f(x_{min}) = f_{min}$$

The algorithm also uses an indicator I_k, which is set to 1 if f(x) is convex in an interval spanning l_k and which is set to 0 otherwise. We put $I_1 = 1$, $I_k = 0$ for $1 \leq k \leq K - 1$. Set $j = K - 1$ and enter a general step of the algorithm, which is as follows: Compute

$$M_1 = \sum_i \min[\pi_i g_i''(l_j),\ \pi_i g_i''(u)]$$

$$M_2 = \sum_i \max[\pi_i g_i''(l_j),\ \pi_i g_i''(u)]$$

If $M_2 \leq 0$ go to (*)

(*) come here if no appreciably better solution can exist for $l_j < x < u$. We then put $u = l_j$. If $j = 1$ then whole problem is solved and $f_{min} = f(x_{min})$ is within ϵ of the minimum value of f(x). Otherwise put $j = j - 1$ and start a new step of the algorithm.

Otherwise proceeds as follows: Compute

$$h = u - l_j,$$

$$t = f'(l_j) + M_1 h$$

$$a = f'(u) - t,$$

$$b = f(u) - f(l_j) - h(f'(l_j) + t)/2$$

If $ah < \epsilon$, then if $M_1 \leq 0$ or if $f'(l_j) \geq 0$, or $t \leq 0$, go to (*). Otherwise put

$x_T = -f'(l_j)/M_1$

$f_T = f(l_j) + f'(l_j)/2 + x_T$

If $ah \geq \epsilon$, compute

$y_1 = a/(M_2 - M_1)$, $\quad x_1 = h - b/a - y_1/2$

$c = f'(l_j) + M_1 x_1$

then if $c \geq 0$ and either $M_1 \leq 0$ or $f'(l_j) \geq 0$ go to (*).

If $c \geq 0$ otherwise, put $x_T = -f'(l_j)/M_1$, $f_T = f(l_j) + f'(l_j)x_T/2$.

If $c \leq -M_2 y_1$ and either $M_1 \leq 0$ or $f''(u) \leq 0$ go to (*).

If $c \leq -M_2 y_1$ otherwise, put

$x_T = u - f'(u)/M_1$, $f_T = f(u) - \{f'(u)\}^2/2M_1$

If $-M_2 y_1 < c < 0$, put

$x_T = x_1 - c/M_2$, $\quad f_T = f(l_j) + (f'(l_j)x_1 + c\, x_T)/2$

If $f_T > f_{min} - \epsilon$, go to (*).

Otherwise put $j = j + 1$, $L_j = l_{j-1} + x_T$ and compute $f(l_j)$, $f'(l_j)$ and $g''_i(l_j)$. If $f(l_j) < f_{min}$, put $f_{min} = f(l_j)$ and $x_{min} = l_j$.

If $M_1 < 0$, put $I_j = 0$ and start a new step of the algorithm.

If $M_1 \geq 0$ and $f'(l_j) \geq 0$ go to (*).

If $M_1 \geq 0$ and $f'(l_j) < 0$, put $I_j = 1$; and if $I_{j-1} = 1$, put $l_{j-1} = l_j$,

$f(l_{j-1}) = f(l_j)$, $g''_i(l_{j-1}) = g''_i(l_j)$ and $j = j - 1$

Start a new step of the algorithm

For the ventilation problem discussed, the function D_{I_j} defining as

$$D_{I_j}(x_j) = \sum_i \max\{P^+_i(g_{ij}(x_j) - g_{I_{ij}}(x_j)), -P^-_i(g_{ij}(x_j) - g_{I_{ij}}(x_j)),$$
$$\pi_i(g_{ij}(x_j) - g_{I_{ij}}(x_j))\}.$$

can be calculated for the functions with respect to the first law constraints, objective, and power constraints.

For the first law constraints,

$$g_j(Q_j) = g_j((1-\theta)Q_{js} + \theta Q_{je}) = (1-\theta)Q_{js} + \theta Q_{je}$$

$$g_{I_j}(Q_j) = (1-\theta)g_j(Q_{js}) + \theta g_j(Q_{je}) = (1-\theta)Q_{js} + \theta Q_{je}$$

where $g_j(Q_j) = Q_j$ for the first law constraints' function.

So, $g_j(Q_j) - g_{I_j}(Q_j) = 0$

For the objective function and power constraints,

$$g_j(Q_j) = g_j((1-\theta)Q_{js} + \theta Q_{je}) = HF_j((1-\theta)Q_{js} + \theta Q_{je})$$

$$\begin{aligned} g_{I_j}(Q_j) &= (1-\theta)g_j(Q_{js}) + \theta g_j(Q_{je}) = (1-\theta)Q_{js}HF_j + \theta Q_{je}HF_j \\ &= HF_j((1-\theta)Q_{js} + \theta Q_{je}) \end{aligned}$$

so, $g_j(Q_j) - g_{I_j}(Q_j) = 0$

It indicates that calculation of function D_{I_j} is carried out only for the functions with respect to the second law when it needs to find out where to branch on.

For the function with respect to second law constraints, the function D_{I_j} can be calculated as

$$g_j(Q_j) = g_j((1-\theta)Q_{js} + \theta Q_{je}) = R_j((1-\theta)Q_{js} + \theta Q_{je})^2$$

$$g_{I_j}(Q_j) = (1-\theta)g_j(Q_{js}) + \theta g_j(Q_{je}) = R_j(1-\theta)Q_{js}^2 + R_j\theta Q_{je}^2$$

$$g_j(Q_j) - g_{I_j}(Q_j) = R_j((1-\theta)Q_{js} + \theta Q_{je})^2 - R_j(1-\theta)Q_{js}^2 - R_j\theta Q_{je}^2$$
$$= R_j((1-\theta)Q_{js} + \theta Q_{je})^2 - (1-\theta)Q_{js}^2 - \theta Q_{je}^2) = -R_j(Q_{je}-Q_{js})^2\theta(1-\theta)$$

where $(1 - \theta)Q_{js} + \theta Q_{je} \geq 0$. Since this is a second order function of θ, the minimization occurs at the middle point.

APPENDIX E

COMPUTER PROGRAM

```
/*This computer program for optimization of ventilation */
/*control device location and size in underground       */
/*ventilation systems                                    */
/*This part of program is used for executive control    */

#include <process.h>                    /* for exit() */
#include <stdlib.h>                     /* for abs()  */
#include <stdio.h>                      /* for printf() */
#include <conio.h>                      /* for getch() */
#define NORMAL "\x1B[0m"                /* normal attribute */
#define REVERSE "\x1B[7m"     /* reverse video attribute */
void display(char **, int, int);
int getcode(void);
void action(int);
void new(void);
void old (void);
void run (void);
main()
{
static char *items[4]=
{"INPUT DATA",
 "EDIT DATA",
 "RUN",
 "QUIT"};
int code;
int curpos=0;
printf("\x1B[2J");
printf("\x1B[5;14f");
printf("OPTIMIZATION OF VENTILATION CONTROL DEVICE");
printf("\x1B[6;16f");
printf("LOCATIONS AND SIZES IN MINE VENTILATION SYSTEMS");
printf("\x1B[12;30f");
printf("Written by Xing Wu");
printf("\x1B[14;18f");
printf("Department of Mining & Minerals Engineering");
printf("\x1B[15;15f");
printf("Virginia Polytechnic Institute & State University");
printf("\x1B[16;30f");
printf("Blacksburg, VA 24061");
printf("\x1B[17;36f");
printf("(1991)");
printf("\x1B[20;2f");
printf("Hint any key to continue...");
if(getch()>=0)
printf("\x1B[2J");
while(1)
{
display(items, 4, curpos);
code=getcode();
```

```
switch(code)
{
case 72:
if(curpos>0) --curpos; break;
case 80:
if(curpos<3) ++curpos; break;
case 13:
action(curpos); break;
defualt:
printf("\x7");
}
}
}                                         /* end of main */
void display(char **arr, int size, int pos)
{
int j;
printf("\x1B[1;1f");
for(j=0; j<size; j++)
{
if (j==pos)
printf(REVERSE);
printf("%s\n",*(arr+j));
printf(NORMAL);
}
printf("\x1B[20;1f");
}
int getcode(void)
{
int key;
if((key=getch())!=0)
return (key);
return (getch());
}
void action(int pos)
{
printf("\x1B[K");
switch (pos)
{
case 0:
new(); break;
case 1:
old(); break;
case 2:
run(); break;
case 3:
exit(0);
}
}
```

```
#define lim 50
#define error 0.25
#include <stdio.h>                    /* defines FILE & NULL */
void airway (void);
void list1(int);
void input1(int);
void last1(int);
void modify1(int);
void last2 (void);
void last3 (void);
void input2 (int);
void store (void);
void airway1 (void);
void order (void);
void mesh (void);
void search (int,int,int,int *,int *,int *,int *);
void form (void);
void coef (void);

int b,n,m;
struct branch
{
int init, final;
float resist, flow, upper, lower, natu, fan, fl, fu;
char afan, areg;
} ark[lim];

/* Input data in interface mode */
void new (void)
{
printf ("\x1B[2J");
printf ("\x1B[2;2f");
printf ("Give the number of branches in the network:");
scanf ("%d",b);
printf ("\x1B[3;2f");
printf ("Give the number of nodes in the network:");
scanf ("%d",n);
m=b-n+1;
airway();
}

void airway (void)
{
int i;
printf("%d",m);
for (i=0; i<b; i++)
{
list1(i);
```

```
input1(i);
last1(i);
}
last2();
}

void list1 (int i)
{
printf("\x1B[2J");
printf("\x1B[1;21f");
printf("Enter Branch Data Below for Branch %d\n",i+1);
printf("\n 1. ID # of the Initial Node of This Branch");
printf("\n 2. ID # of the Final Node of This Branch");
printf("\n 3. Resistance Factor");
printf("\n 4. Fixed Flow Rate (if it is free,type '0.00')");
printf("\n 5. Upper Bound of the Flow(if it is fixed,
type 'enter')");
printf("\n 6. Lower Bound of the Flow(if it is fixed,
type 'enter')");
printf("\n 7. Nutural Ventilation Pressure( if unknown,
type '0.00')");
printf("\n 8. Fan Head(if no fan, type '0.00')");
printf("\n 9. Allowable Fan(y/n)?");
printf("\n 10. Minimum Fan Pressure(if not allowed,
type 'enter')");
printf("\n 11. Maximum Fan Pressure(if not allowed,
type 'enter')");
printf("\n 12. Allowable Regulator(y/n)?");
}

void input1 (int i)
{
printf("\x1B[3;70f");
scanf("%d", &ark[i].init);
printf("\x1B[4;70f");
scanf("%d", &ark[i].final);
printf("\x1B[5;70f");
scanf("%f", &ark[i].resist);
printf("\x1B[6;70f");
scanf("%f", &ark[i].flow);
printf("\x1B[7;70f");
scanf("%f", &ark[i].upper);
printf("\x1B[8;70f");
scanf("%f", &ark[i].lower);
printf("\x1B[9;70f");
scanf("%f", &ark[i].natu);
printf("\x1B[10;70f");
scanf("%f", &ark[i].fan);
printf("\x1B[11;70f");
scanf("%c", &ark[i].afan);
```

```
printf("\x1B[12;70f");
scanf("%f", &ark[i].fl);
printf("\x1B[13;70f");
scanf("%f", &ark[i].fu);
printf("\x1B[14;70f");
scanf("%c", &ark[i].areg);
}

void last1(int i)
{
char ch;
int ture=1;
printf("\x1B[20; 1f");
printf("\x1B[k");
printf("Enter 'c' for Next Branch,'e' for Modifying");
ch=getche();
while (ture)
{
switch(ch)
{
case'c':ture=0;break;
case'e':modify1(i); ture=0;break;
default:
printf("You have to type 'c' to continue or 'e' to edit");
ch=getche();
}
}
}

void modify1(i)
{
int ch;
printf("\x1B[20; 1f");
printf("\x1B[K");
printf("Enter the item(1,2,...,12) needed to be modified");
ch=getche();
switch(ch)
{
case 1:
printf("\x1B[3;70f");
scanf("%d", &ark[i].init); last1(i); break;
case 2:
printf("\x1B[4; 70f");
scanf("%d", &ark[i].final); last1(i); break;
case 3:
printf("\x1B[5; 70f");
scanf("%f", &ark[i].resist); last1(i); break;
case 4:
printf("\x1B[6; 70f");
scanf("%f", &ark[i].flow); last1(i); break;
```

```
case 5:
printf("\x1B[7; 70f");
scanf("%f", &ark[i].upper); last1(i); break;
case 6:
printf("\x1B[8; 70f");
scanf("%f", &ark[i].lower); last1(i); break;
case 7:
printf("\x1B[9; 70f");
scanf("%f", &ark[i].natu); last1(i); break;
case 8:
printf("\x1B[10; 70f");
scanf("%f", &ark[i].fan); last1(i); break;
case 9:
printf("\x1B[11: 70f");
scanf("%c", &ark[i].afan); last1(i); break;
case 10:
printf("\x1B[12; 70f");
scanf("%f", &ark[i].fl);   last1(i); break;
case 11:
printf("\x1B[13; 70f");
scanf("%f", &ark[i].fu); last1(i); break;
case 12:
printf("\x1B[14; 70f");
scanf("%c", &ark[i].areg); last1(i); break;
}
}

void last2 (void)
{
char ch;
int ture=0;
printf("\x1B[2J");
printf("\x1B[20,1f");
printf("\x1B[k");
printf("Type 'c' to continue, 'e' to edit");
ch=getche();
while (ture)
{
switch(ch)
{
case'c':
store(); ture=0;break;
case'e':
last3(); ture=0;break;
default:
printf("You have to type the defined characters");
ch=getche();
}
}
}
```

```
void last3 (void)
{
int i,ch;
printf("\x1B[20;1f");
printf("\x1B[K");
printf("Give # of the branch needed to be modified");
ch=getche();
i=ch-1;
list1(i);
input2 (i);
modify1(i);
last2 ();
}

void input2 (int i)
{
printf("\x1B[3;70f");
printf("%d",ark[i].init);
printf("\x1B[4;70f");
printf("%d",ark[i].final);
printf("\x1B[5;70f");
printf("%f", ark[i].resist);
printf("\x1B[6;70f");
printf("%f", ark[i].flow);
printf("\x1B[7;70f");
printf("%f",ark[i].upper);
printf("\x1B[8;70f");
printf("%f",ark[i].lower);
printf("\x1B[9;70f");
printf("%f", ark[i].natu);
printf("\x1B[10;70f");
printf("%f",ark[i].fan);
printf("\x1B[11;70f");
printf("%c",ark[i].afan);
printf("\x1B[12;70f");
printf("%f",ark[i].fl);
printf("\x1B[13;70f");
printf("%f",ark[i].fu);
printf("\x1B[14;70f");
printf("%c",ark[i].areg);
}

void store (void)
{
char fname[20];
FILE *fptr;
printf("\x1B[2J");
printf("\x1B[2;2f");
printf("\nGive Data File Name:");
```

```
gets(fname);
if((fptr=fopen("fname","wb"))==NULL)
{printf("Can't open file %s",fname); exit();}
fwrite(ark,sizeof(ark[0]),b,fptr);
fclose(fptr);
}

/*Read in data from data file and review/modify them */
void old (void)
{
char ch, name[20];
FILE *fptr1;
printf ("\x1B[2J");
printf ("\x1B[2;2f");
printf ("Give the number of branches in the network:");
scanf ("%d",b);
printf ("\x1B[3;2f");
printf ("Give the number of nodes in the network:");
scanf ("%d",n);
m=b-n+1;
printf ("\x1B[4;2f");
printf ("Give data file name:");
gets (name);
if ((fptr1=fopen("name","rb"))==NULL)
{printf("Can't open file %s",name); exit(); }
fread (ark, sizeof(ark[0]),b,fptr1);
fclose (fptr1);
printf ("\x1B[2J");
printf ("\x1B[20;1f");
printf ("Type 'c' to review & modify branch data,
'r' to calculate");
ch =getche();
switch (ch)
{
case 'c':
airway1();break;
case 'r':
break;
}
}

void airway1 (void)
{
int i;
for (i=0;i<b;i++)
{
list1(i);
input2(i);
last1(i);
}
```

```
last2();
}

void run (void)
{
order ();
mesh ();
form ();
coef ();
}

/* Determine chord branches */
int chord[lim/2];
void order (void)
{
int i,j,temp,temp1,count;
int bar[lim][4];

for (i=0;i<lim;i++)
{
bar[i][0]=i+1;
for (j=1;i<4;j++)
bar[i][j]=0;
}
for (i=0;i<lim/2;i++)
chord[i]=0;
for (i=0;i<b-1;i++)
for (j=i+1;j<b; j++)
if (ark[bar[j][0]-1].resist<ark[bar[i][0]-1].resist)
{
temp=bar[i][0];
bar[i][0]=bar[j][0];
bar[j][0]=temp;
}
bar[0][1]=bar[0][2]=bar[0][3]=count=1;
for (i=1;i<b;i++)
{
for (j=0;j<i;j++)
{
if (ark[bar[i][0]-1].init==ark[bar[j][0]-1].init)
bar[i][1]=bar[j][1];
if (ark[bar[i][0]-1].init==ark[bar[j][0]-1].final)
bar[i][1]=bar[j][2];
if (ark[bar[i][0]-1].final==ark[bar[j][0]-1].init)
bar[i][2]=bar[j][1];
if (ark[bar[i][0]-1].final==ark[bar[j][0]-1].final)
bar[i][2]=bar[j][2];
}
if (bar[i][1]==bar[i][2]&&bar[i][1]!=0)
bar[i][3]=2;
else
if (bar[i][1]==0&&bar[i][2]==0)
{
bar[i][1]=bar[i][2]=++count;
bar[i][3]=1;
}
else
if (bar[i][1]=0)
{
bar[i][1]=bar[i][2];
bar[i][3]=1;
}
else
if (bar[i][2]=0)
{
bar[i][2]=bar[i][1];
bar[i][3]=1;
}
else
{
bar[i][3]=1;
temp=(bar[i][1]>bar[i][2])?bar[i][1]:bar[i][2];
temp1=(bar[i][1]<bar[i][2])?bar[i][1]:bar[i][2];
bar[i][1]=bar[i][2]=temp1;
for (j=0;j<i;j++)
{
if (bar[j][1]==temp)
bar[j][1]=temp1;
if (bar[j][2]==temp)
bar[j][2]=temp1;
}
if (count>temp)
for (j=1;j<i;j++)
{
if (bar[j][1]>temp)
bar[j][1]=bar[j][1]-1;
if (bar[j][2]>temp)
bar[j][2]=bar[j][2]-1;
}
count--;
}
}
temp=0;
if (count==1)
{
for (i=0;i<b;i++)
if (bar[i][3]==2)
chord[temp++]=bar[i][0];
}
else
```

```
{
printf("\x1B[2J");
printf("\x1B[2;20f");
printf("No spanning tree exists.");
exit();
}
for (i=0;i<b-1;i++)
for (j=i+1;j<b;j++)
if (chord[j]<chord[i])
{
temp=chord[i];
chord[i]=chord[j];
chord[j]=temp;
}
}                                         /* end of order */

/* Form the fundamental mesh of the network */
int cycle[lim/2][lim];
void mesh (void)                  /* form the mesh matrix*/
{
int i,k,j,l,line,ture,start,end;
int sign[lim],seq[lim*3/4];
for (j=0; j<lim/2; j++)
for (k=0; k<lim; k++)
{
cycle[j][k]=0;
sign[k]=0;
}
for (i=0;i<lim*3/4;i++)
seq[i]=0;
for (i=0; i<m; i++)
{
end=ark[chord[i]-1].init;
for (k=0; k<b; k++)
sign[k]=0;
for (k=1; k<b-m; k++)
seq[k]=0;
seq[0]=chord[i];
ture=l=0;
while (ture!=1)
{
line=seq[0];
if ( line==chord[i])
start=ark[line-1].final;
else
if (sign[line-1]==1)
start=ark[line-1].final;
else
start=ark[line-1].init;
search(line, start,end, &l,&ture,sign,seq);
for (k=0; k<l-1; k++)
seq[k]=seq[k+1];
l=l-1;
}                                        /* end of while */
}                                        /* end of for   */
}                                        /* end of mesh  */

void search (int line, int start,int end, int *l,int *ture,
                                          int *sign,int *seq)
{
int i,k, temp, count, j=-1;
int back[lim];
for (k=0;k<lim;k++)
back[k]=0;
while (j++<b)
{
count=0;
if((ark[j].init==start&&j+1!=line)
||(ark[j].final==start&&j+1!=line))
{
for (k=0;k<m;k++)
if (j+1==chord[k])
{
count=1;
break;
}
if( count==0)
{
if (ark[j].init==start)
*(sign+j)=1;
else
*(sign+j)=-1;
*l=*l+1;
*(seq+*l)=j+1;
back[j]=line;
if (ark[j].final==end||ark[j].init==end)
{
*ture=1;
temp=j+1;
while (temp!=chord[i])
{
cycle[i][temp-1]=sign[temp];
temp=back[temp];
}
cycle[i][chord[i]-1]=1;
j=b;
}                                         /* end of if    */
}                                         /* end of if    */
}                                         /* end of if    */
}                                         /* end of while */
}                                       /* end of search */

/Calculate the first and second laws' matrix */
int q2[lim*3/4],q12[lim*3/4][lim/2],q1r[lim*3/4];
int chord1[lim/2];
float q2r[lim*3/4];
int nft,nre,ndf,nset,nqh,nst,nfs,nfsf;
void form (void)                  /* prepare for first law*/
{
int k,j,i,l,count,temp;
int b12[lim/2][lim*3/4],b12t[lim*3/4][lim/2];
float constant;
nft=nre=ndf=nset=nfs=nst=nqh=nfsf=0;
for (i=0; i<lim*3/4;i++)
{
q1r[i]=q2r[i]=q2[i]=0;
for (j=0; j<lim/2; j++)
q12[i][j]=b12[j][i]=b12t[i][j]=0;
}
for (i=0; i<m; i++)
{
k=l=0;
for (j=0; j<b; j++)
if (j+1==chord[k])
k++;
else
{
b12[i][l]=cycle[i][j];
q2[l++]=j+1;
}
if (ark[chord[i]-1].flow==0)
{
nset++;
if (ark[chord[i]-1].afan=='y')
nqh++;
}
}
for (i=0;i<b-m;i++)
{
count=constant=temp=0;
for (j=0;j<m;j++)
{
b12t[i][j]=b12[j][i];
q12[i][j]=b12t[i][j]*chord[j];
if (q12[i][j]!=0)
if(ark[abs(q12[i][j])-1].flow==0)
{
count++;
temp=q12[i][j];
}
```

```
else
{
a=q12[i][j]/abs(q12[i][j])*ark[abs(q12[i][j])-1].flow;
constant+=a;
}
}
if (count==0)
q1r[i]=0;
else
if(count>1)
{
q1r[i]=q2[i];
if (ark[q2[i]-1].afan=='y')
nqh++;
}
else
q1r[i]=temp;
q2r[i]=constant;
if (q1r[i]==0&&ark[q2[i]-1].flow==0)
ark[q2[i]-1].flow=q2r[i];        /* updata struct branch */
if (q1r[i]==q2[i])
{
nset++;
nft++;
}
}                                                   /* end of for */
for (i=0;i<b;i++)
{
if (ark[i].areg=='y')
nre++;
if (ark[i].afan=='y'&&ark[i].flow!=0)
ndf++;
}
for (i=0;i<m;i++)
{
chord1[i]=0;
if (ark[chord[i]-1].flow==0)
for (j=0;j<b-m;j++)
if (q1r[j]!=q2[j]&&abs(q1r[j])==chord[i])
{
nfsf++;
if (chord1[i]==0)
chord1[i]=q2[j];
else
chord1[i]=chord1[i]*1000+q2[j];
if (ark[q2[j]-1].afan=='y')
{
nfs++;
if (ark[chord[i]-1].afan=='n')
nst++;
}
}
}
}                                                 /* end of form */
```

```
#define lim 50
#include <stdio.h>
extern int nft,nre,ndf,nset,nqh,nst,nfs,nfsf,m,b;
extern int q2[lim*3/4],q1r[lim*3/4],q12[lim*3/4][lim/2];
extern int chord[lim/2],chord1[lim/2],cycle[lim*3/4][lim];
extern float q2r[lim*3/4];
extern struct branch
{
int init,final;
float resist,flow,upper,lower,natu,fan,fl,fu;
char afan,areg;
} ark [lim];

void cher (int *, int *,int, int);
void instd (float *,int *,float *,int *,int *,int *,int *,
int,int);
void pw (int *,float *, int, float,float,float);
void obj (int ,float *,int ,float , float ,float);
void obpw (float *,int [],float [],float [],float []);
void cont (int *,int *,float *,int);
void obfs (int *,float *,float *,float *,int,float);
float coefb (int);
extern void fortran user(int,int,int,int,int,int,int,int,
                         int,int,float,char inf[]);

/* Calculate the coefficients for 1st LP model variables */
void coef (void)
{
FILE *fptr;
int i,j,k,l,count,count1,temp,temp1,nb;
int dfi[lim/2],rei[lim/2],set[lim/2],set1[lim/2];
int set2[lim/2],isff[lim/2],sfi[lim/2];
float ftsc[lim*3],constant,rest[lim/2];
float resf[lim/2],setpt1[lim/2],setpt2[lim/2],setpt3[lim/2];
i=j=k=l=temp=temp1=count=count1=constant=0;
for (i=0;i<lim/2;i++)
{
set[i]=set2[i]=rest[i]=set1[i]=dfi[i]=rei[i]=0;
setpt1[i]=setpt2[i]=setpt3[i]=resf[i]=0;
sfi[i]=isff[i]=0;
}
for (i=0;i<lim*3;i++)
ftsc[i]=0;
if ((fptr=fopen("cpass.dat","wb"))==NULL)
{printf("can't open file  cpass.dat"); exit();}
j=0;
for (i=0;i<m;i++)
if (ark[chord[i]-1].flow==0)
{
set[j]=chord[i];
set2[j++]=chord1[i];
}
for (i=0;i<b-m;i++)
if (q2[i]==q1r[i])
{
set[j]=q2[i];
set2[j++]=0;
}
for (i=1;i<nset;i++)
if (ark[set[i]-1].afan=='y')
for (j=0;j<i;j++)
if (ark[set[j]-1].afan=='n')
{
temp=set[i];
temp1=set2[i];
set[i]=set[j];
set2[i]=set2[j];
set[j]=temp;
set2[j]=temp1;
break;
}
for (i=0;i<nset;i++)
for (k=0;k<b-m;k++)
if (q2[k]!=q1r[k])
if (ark[q2[k]-1].afan=='y'&&set[i]==abs(q1r[k]))
set1[i]=1;
for (i=nqh+1;i<nset;i++)
if (set1[i]==1)
for (j=nqh;j<i;j++)
if (set1[j]==0)
{
temp=set[i];
temp1=set2[i];
set[i]=set[j];
set2[i]=set2[j];
set[j]=temp;
set2[j]=temp1;
break;
}
cher (set,set2, 0, nqh);
cher (set,set2, nqh, nqh+nst);
cher (set,set2, nqh+nst, nset);
for (i=0;i<nset;i++)
{
k=set[i]-1;
setpt1[i]=ark[k].lower;
setpt2[i]=(ark[k].lower+ark[k].upper)/2;
setpt3[i]=ark[k].upper;
rest[i]=ark[k].resist;
}
```

```
fwrite(set,sizeof(set[0]),nset,fptr);
fwrite(set2,sizeof(set2[0]),nset,fptr);
fwrite(setpt1,sizeof(setpt1[0]),nset,fptr);
fwrite(setpt2,sizeof(setpt2[0]),nset,fptr);
fwrite(setpt3,sizeof(setpt3[0]),nset,fptr);
fwrite(rest,sizeof(rest[0]),nset,fptr);
j=l=0;
for (i=0;i<b;i++)
{
if (ark[i].flow!=0&&ark[i].afan=='y')
{
dfi[j]=i+1;
rest[j]=ark[i].flow;
resf[j++]=ark[i].fl;
}
if (ark[i].areg=='y')
rei[l++]=i+1;
}
fwrite(rei,sizeof(rei[0]),nre,fptr);
if (ndf>0)
{
fwrite(rest,sizeof(rest[0]),ndf,fptr);
fwrite(resf,sizeof(resf[0]),ndf,fptr);
fwrite(dfi,sizeof(dfi[0]),ndf,fptr);
}
j=l=0;
for (i=0;i<nset;i++)
if (set2[i]>0&&set2[i]<1000)
instd (rest,set1,resf,isff,sfi,&j,&l,set2[i],i);
else
if (set2[i]>1000&&set2[i]<1000000)
{
temp=set2[i]/1000;
instd (rest,set1,resf,isff,sfi,&j,&l,temp,i);
temp=set2[i]-temp*1000;
instd (rest,set1,resf,isff,sfi,&j,&l,temp,i);
}
else
if (set2[i]>1000000)
{
temp=set2[i]/1000000;
instd (rest,set1,resf,isff,sfi,&j,&l,temp,i);
temp1=(set2[i]-temp*1000000)/1000;
instd (rest,set1,resf,isff,sfi,&j,&l,temp1,i);
temp=set2[i]-temp*1000000-temp1*1000;
instd (rest,set1,resf,isff,sfi,&j,&l,temp,i);
}
if (nfs>0)
fwrite(sfi,sizeof(sfi[0]),nfs,fptr);
if (nfsf>0)
```

```
{
fwrite(set1,sizeof(set1[0]),nfsf,fptr);
fwrite(isff,sizeof(isff[0]),nfsf,fptr);
fwrite(rest,sizeof(rest[0]),nfsf,fptr);
fwrite(resf,sizeof(resf[0]),nfsf,fptr);
}
count=0;                                  /* first law */
for (i=0;i<b-m;i++)
if (q1r[i]==q2[i])
{
rest[count++]=q2r[i];
for (j=0;j<nset;j++)
{
temp1=0;
if (j<nqh+nst)
temp=j*6;
else
temp=(j+nqh+nst)*3;
if (set[j]==q2[i])
{
temp1=1;
l=1;
}
else
for (k=0;k<m;k++)
if (set[j]==abs(q12[i][k]))
{
temp1=1;
l=-q12[i][k]/abs(q12[i][k]);
break;
}
if (temp1==1)
{
if (j<nqh+nst)
{
ftsc[temp+3]=l*setpt1[j];
ftsc[temp+4]=l*setpt2[j];
ftsc[temp+5]=l*setpt3[j];
}
ftsc[temp]=l*setpt1[j];
ftsc[temp+1]=l*setpt2[j];
ftsc[temp+2]=l*setpt3[j];
}
}
fwrite(ftsc,sizeof(ftsc[0]),(nqh+nst+nset)*3,fptr);
}
if (nft>0)
fwrite(rest,sizeof(rest[0]),nft,fptr); /* end of 1st law */
for (i=0;i<nqh;i++)                           /* obj,pw,ref */
rest[i]=ark[set[i]-1].fl*setpt1[i];
```

```
obpw (ftsc,set,setpt1,setpt2,setpt3);
fwrite(rest,sizeof(rest[0]),nqh,fptr);
fwrite(ftsc,sizeof(ftsc[0]),nqh*6,fptr);
for (i=0;i<nqh;i++)
{
j=i*6;
ftsc[j]=ftsc[j+1]=ftsc[j+2]=ark[set[i]-1].fl;
ftsc[j+3]=ftsc[j+4]=ftsc[j+5]=ark[set[i]-1].fu;
}
fwrite(ftsc,sizeof(ftsc[0]),nqh*6,fptr);
count=-1;
for (i=0;i<nqh+nst;i++)
if (set2[i]>0&&set2[i]<1000)
{
if (ark[set2[i]-1].afan=='y')
obfs (&count,resf,rest,ftsc,set2[i],setpt1[i]);
}
else
if (set2[i]>1000&&set2[i]<1000000)
{
temp=set2[i]/1000;
if (ark[temp-1].afan=='y')
obfs (&count,resf,rest,ftsc,temp,setpt1[i]);
temp=set2[i]-temp*1000;
if (ark[temp-1].afan=='y')
obfs (&count,resf,rest,ftsc,temp,setpt1[i]);
}
else
if (set2[i]>1000000)
{
temp=set2[i]/1000000;
if (ark[temp-1].afan=='y')
obfs (&count,resf,rest,ftsc,temp,setpt1[i]);
temp1=(set2[i]-temp*1000000)/1000;
if (ark[temp1-1].afan=='y')
obfs (&count,resf,rest,ftsc,temp1,setpt1[i]);
temp=set2[i]-temp*1000000-temp1*1000;
if (ark[temp-1].afan=='y')
obfs (&count,resf,rest,ftsc,temp,setpt1[i]);
}
if (nfs>0)
{
fwrite(rest,sizeof(rest[0]),nfs,fptr);
fwrite(resf,sizeof(resf[0]),nfs,fptr);
fwrite(ftsc,sizeof(ftsc[0]),nfs*6,fptr);
}
count=-1;
for (i=0;i<nqh+nst;i++)
if (set2[i]>0&&set2[i]<1000)
{
```

```
if (ark[set2[i]-1].afan=='y')
pw (&count,ftsc,set2[i],setpt1[i],setpt2[i],setpt3[i]);
}
else
if (set2[i]>1000&&set2[i]<1000000)
{
temp=set2[i]/1000;
if (ark[temp-1].afan=='y')
pw (&count,ftsc,temp,setpt1[i],setpt2[i],setpt3[i]);
temp=set2[i]-temp*1000;
if (ark[temp-1].afan=='y')
pw (&count,ftsc,temp,setpt1[i],setpt2[i],setpt3[i]);
}
else
if (set2[i]>1000000)
{
temp=set2[i]/1000000;
if (ark[temp-1].afan=='y')
pw (&count,ftsc,temp,setpt1[i],setpt2[i],setpt3[i]);
temp1=(set2[i]-temp*1000000)/1000;
if (ark[temp1-1].afan=='y')
pw (&count,ftsc,temp1,setpt1[i],setpt2[i],setpt3[i]);
temp=set2[i]-temp*1000000-temp1*1000;
if (ark[temp-1].afan=='y')
pw (&count,ftsc,temp,setpt1[i],setpt2[i],setpt3[i]);
}
if (nfs>0)
fwrite(ftsc,sizeof(ftsc[0]),nfs*6,fptr);
obpw (ftsc,set,setpt1,setpt2,setpt3);
for (i=0;i<nqh+nst;i++)
{
j=i*6;
if (set2[i]>0&&set2[i]<1000)
{
if (ark[set2[i]-1].afan=='y')
obj (j,ftsc,set2[i],setpt1[i],setpt2[i],setpt3[i]);
}
else
if (set2[i]>1000&&set2[i]<1000000)
{
temp=set2[i]/1000;
if (ark[temp-1].afan=='y')
obj (j,ftsc,temp,setpt1[i],setpt2[i],setpt3[i]);
temp=set2[i]-temp*1000;
if (ark[temp-1].afan=='y')
obj (j,ftsc,temp,setpt1[i],setpt2[i],setpt3[i]);
}
else
if (set2[i]>1000000)
{
```

```
temp=set2[i]/1000000;
if (ark[temp-1].afan=='y')
obj (j,ftsc,temp,setpt1[i],setpt2[i],setpt3[i]);
temp1=(set2[i]-temp*1000000)/1000;
if (ark[temp1-1].afan=='y')
obj (j,ftsc,temp1,setpt1[i],setpt2[i],setpt3[i]);
temp=set2[i]-temp*1000000-temp1*1000;
if (ark[temp-1].afan=='y')
obj (j,ftsc,temp,setpt1[i],setpt2[i],setpt3[i]);
}
}
fwrite(ftsc,sizeof(ftsc[0]),(nqh+nst)*6,fptr);
for (i=0;i<m;i++)                              /* second law */
{
for (k=0;k<nre;k++)
set1[k]=0;
for (k=0;k<ndf;k++)
isff[k]=0;
for (j=0;j<b;j++)
if (cycle[i][j]!=0)
{
if (ark[j].areg=='y')
for (k=0;k<nre;k++)
if (rei[k]==j+1)
{
set1[k]=cycle[i][j];
break;
}
if (ark[j].afan=='y'&&ark[j].flow!=0)
for (k=0;k<ndf;k++)
if (dfi[k]==j+1)
{
isff[k]=-cycle[i][j];
break;
}
}
fwrite(set1,sizeof(set1[0]),nre,fptr);
fwrite(isff,sizeof(isff[0]),ndf,fptr);
}
for (i=0;i<m;i++)
{
for (k=0;k<nqh;k++)
set1[k]=0;
for (k=0;k<nfs;k++)
isff[k]=0;
for (j=0;j<b;j++)
if (cycle[i][j]!=0)
if (ark[j].afan=='y'&&ark[j].flow==0)
{
for (k=0;k<nqh;k++)
```

```
if (set[k]==j+1)
{
set1[k]=-cycle[i][j];
break;
}
for (k=0;k<nfs;k++)
if (sfi[k]==j+1)
{
isff[k]=-cycle[i][j];
break;
}
}
fwrite(set1,sizeof(set1[0]),nqh,fptr);
fwrite(isff,sizeof(isff[0]),nfs,fptr);
}
for (i=0;i<m;i++)
{
for (k=0;k<m;k++)
rest[k]=0;
for (k=0;k<(nset+nqh+nst)*3;k++)
ftsc[k]=0;
for (j=0;j<b;j++)
if (cycle[i][j]!=0)
{
if (ark[j].natu!=0)
rest[i]+=cycle[i][j]*ark[j].natu;
if (ark[j].fan!=0)
rest[i]+=cycle[i][j]*ark[j].fan;
if (ark[j].flow!=0)
rest[i]+=-cycle[i][j]*ark[j].resist*abs(ark[j].flow)
          *ark[j].flow;
else
for (k=0;k<nset;k++)
{
count1=nb=0;
constant=0;
if (k<nqh+nst)
count=k*6;
else
count=(k+nqh+nst)*3;
if (set[k]==j+1)
{
count1=1;
nb=set[k];
constant=0;
}
else
if (set2[k]>0&&set2[k]<1000)
{
if (set2[k]==j+1)
cont (&count1,&nb,&constant,set2[k]);
}
else
if (set2[k]>1000&&set2[k]<1000000)
{
temp=set2[k]/1000;
if (temp==j+1)
cont (&count1,&nb,&constant,temp);
temp=set2[k]-temp*1000;
if (temp==j+1)
cont (&count1,&nb,&constant,temp);
}
else
if (set2[k]>1000000)
{
temp=set2[k]/1000000;
if (temp==j+1)
cont (&count1,&nb,&constant,temp);
temp1=(set2[k]-temp*1000000)/1000;
if (temp1==j+1)
cont (&count1,&nb,&constant,temp1);
temp=set2[k]-temp*1000000-temp1*1000;
if (temp==j+1)
cont (&count1,&nb,&constant,temp);
}
if (count1==1)
{
if (k<nqh+nst)
{
ftsc[count+3]+=ark[nb-1].resist*abs(setpt1[k]+constant)
                  *(setpt1[k]+constant);
ftsc[count+4]+=ark[nb-1].resist*abs(setpt2[k]+constant)
                  *(setpt2[k]+constant);
ftsc[count+5]+=ark[nb-1].resist*abs(setpt3[k]+constant)
                  *(setpt3[k]+constant);
}
ftsc[count]+=ark[nb-1].resist*abs(setpt1[k]+constant)
                  *(setpt1[k]+constant);
ftsc[count+1]+=ark[nb-1].resist*abs(setpt2[k]+constant)
                  *(setpt2[k]+constant);
ftsc[count+2]+=ark[nb-1].resist*abs(setpt3[k]+constant)
                  *(setpt3[k]+constant);
}
}
}
fwrite(ftsc,sizeof(ftsc[0]),(nqh+nst+nset)*3,fptr);
}
fwrite(rest,sizeof(rest[0]),m,fptr);   /* end of second */
for (i=0;i<m;i++)
{
```

```
for (j=0;j<b;j++)
if (cycle[i][j]!=0)
{
ftsc[j]=0;
if (ark[j].flow==0)
ftsc[j]=1;
if (ark[j].areg=='y')
ftsc[j]+=10;
if (ark[j].afan=='y')
ftsc[j]+=100;
}
fwrite (ftsc,sizeof(ftsc[0]), b,fptr);
}
fclose(fptr);
user (m,b,nft,ndf,nre,nset,nqh,nst,nfs,nfsf,error,infile);
}                                             /* end of coef */

void cher (int *ptr, int *ptr1, int n, int m)
{
int i, j, temp, temp1;
for (i=n;i<m-1;i++)
for (j=i+1;j<m;j++)
if (*(ptr+j)<*(ptr+i))
{
temp=*(ptr+i);
temp1=*(ptr1+i);
*(ptr+i)=*(ptr+j);
*(ptr1+i)=*(ptr1+j);
*(ptr+j)=temp;
*(ptr1+j)=temp1;
}
}

float coefb (int count)
{
int i;
float constant=0;
for (i=0; i<b-m; i++)
if (count==q2[i])
{
constant=q2r[i];
break;
}
return (constant);
}

void pw(int *ptr1,float *ptr,int n,float pt1,
        float pt2,float pt3)
{
int j;
```

```
*ptr1=*ptr1+1;
j=*ptr1*6;
*(ptr+j)=ark[n-1].fl*pt1;
*(ptr+j+1)=ark[n-1].fl*pt2;
*(ptr+j+2)=ark[n-1].fl*pt3;
*(ptr+j+3)=ark[n-1].fu*pt1;
*(ptr+j+4)=ark[n-1].fu*pt2;
*(ptr+j+5)=ark[n-1].fu*pt3;
}

void obj (int j,float *ptr,int n,float pt1,
          float pt2,float pt3)
{
*(ptr+j)+=ark[n-1].fl*pt1;
*(ptr+j+1)+=ark[n-1].fl*pt2;
*(ptr+j+2)+=ark[n-1].fl*pt3;
*(ptr+j+3)+=ark[n-1].fu*pt1;
*(ptr+j+4)+=ark[n-1].fu*pt2;
*(ptr+j+5)+=ark[n-1].fu*pt3;
}

void instd (float *ptr1,int *ptr2,float *ptr3,int *ptr4,
            int *ptr5, int *j,int *l, int temp, int i)
{
int k;
*(ptr1+*j)=ark[temp-1].resist;
for (k=0;k<b-m;k++)
if (q2[k]==temp)
{
if (q1r[k]<0)
*(ptr2+*j)=-i;
else
*(ptr2+*j)=i;
*(ptr3+*j)=q2r[k];
break;
}
*(ptr4+*j)=temp;
*j=*j+1;
if (ark[temp-1].afan=='y')
*(ptr5+*l)=temp;
*l=*l+1;
}

void cont (int *ptr1,int *ptr2,float *ptr3,int temp)
{
*ptr1=1;
*ptr2=temp;
*ptr3=coefb(temp);
}
```

```
void obfs (int *ptr1,float *ptr2,float *ptr3,float *ptr4,
           int temp, float constant)
{
int k;
*ptr1=*ptr1+1;
k=*ptr1*6;
*(ptr2+*ptr1)=ark[temp-1].fl*constant;
*(ptr3+*ptr1)=coefb(temp);
*(ptr4+k)=*(ptr4+k+1)=*(ptr4+k+2)=ark[temp-1].fl;
*(ptr4+k+3)=*(ptr4+k+4)=*(ptr4+k+5)=ark[temp-1].fu;
}

void obpw (float *ptr,int set[],float setpt1[],
           float setpts[],float setpt3[])
{
int i,j;
for (i=0;i<nqh;i++)
{
j=i*6;
*(ptr+j)=ark[set[i]-1].fl*setpt1[i];
*(ptr+j+1)=ark[set[i]-1].fl*setpt2[i];
*(ptr+j+2)=ark[set[i]-1].fl*setpt3[i];
*(ptr+j+3)=ark[set[i]-1].fu*setpt1[i];
*(ptr+j+4)=ark[set[i]-1].fu*setpt2[i];
*(ptr+j+5)=ark[set[i]-1].fu*setpt3[i];
}
}
```

```
      SUBROUTINE USER(M,B,NF,ND,NE,N,NH,NT,NS,NSF,ER,INFILE)
      INTEGER M [VALUE]
      INTEGER B [VALUE]
      INTEGER NF [VALUE]
      INTEGER ND [VALUE]
      INTEGER NE [VALUE]
      INTEGER N [VALUE]
      INTEGER NH [VALUE]
      INTEGER NT [VALUE]
      INTEGER NS [VALUE]
      INTEGER NSF [VALUE]
      REAL    ER [VALUE]
      CHARACTER*9 INFILE

C MATRIX GENERATOR FOR INITIAL LINEARIZATION PROBLEM
C    DATA FILE IS INPUT BY C PROGRAM
C    THIS FORTRAN PROGRAM INTERFACES WITH LINDO
      COMMON /USERCM/ NH,NT,NPW,NF,M,ND,NS,NSF
     1 ISF(NS),RESC(NH*6),RESCS(NS*6),REST(N)
     2 FTSC(NF,(N+NH+NT)*3),SDSC(M,(N+NH+NT)*3)
     3 ISET3(N),IRESF(NSF),ISFF(NSF),RESF(NSF,2)
      DIMENSION VNAME(8),IRO(1+NF+M+N+ND+2*NH+2*NS)
      DIMENSION VAL(1+NF+M+N+ND+2*NH+2*NS),IRE(NE),RHSS(M0
      DIMENSION RHSP1(ND),RHSP2(NH),RHP3(NS),RHSF(NF),
      DIMENSION ISDRE(M,NE), IDF(ND), OBDF(ND),ISDDF(M,ND)
      DIMENSION ISDFF(M,NH),OBSC(NH*6+NT*6),PWSC(NH*6)
      DIMENSION PWSCS(NS*6),ISDFS(M,NS),ISET1(N)
      DIMENSION LIST(20),DLIST(20),SET(3),BS(20,N,2),ISOL(N)
      DIMENSION TSOL(N),REPS(NSF),INDEX(3),EXM(N),OBFS(NS)
      DIMENSION SETPTS(N,3),SAVSTS(N,3),ICYC(M,B)
      CHARACTER*4 CA(N),CC(NSF)
      LOGICAL  TRUBLE
C     ALPHABET
      DATA ALFANM/1H0,1H1,1H2,1H3,1H4,1H5,1H6,1H7,1H8,1H9,
     * 1HA,1HB,1HC,1HD,1HE,1HF,1HG,1HH,1HI,1HJ,1HK,1HL,1HM,
     * 1HN,1HO,1HP,1HQ,1HR,1HS,1HT,1HU,1HV,1HW,1HX,1HY,1HZ/
      DATA BLANK /' '/
      DATA I5/5/
C
C     READ IN DATA FROM THE DATA FILE
C
      OPEN(5,FILE=INFILE,STATUS='OLD')
      NPW=NH+NS+ND
      DO 5 I=1,N
      READ (I5,*) ISET(I)
    5 CONTINUE
```

```
   DO 10 I=1,N
   READ (I5,*) ISET3(I)
10 CONTINUE
   DO 15 I=1,3
   DO 15 J=1,N
   READ (I5,*) SETPTS(I,J)
15 CONTINUE
   DO 20 I=1,N
   READ (I5,*) REST(I)
20 CONTINUE
   DO 25 I=1,NE
   READ (I5,*) IRE(I)
25 CONTINUE
   DO 30 I=1,ND
   READ (I5,*) OBDF(I)
30 CONTINUE
   DO 35 I=1,ND
   READ (I5,*) RHSP1
35 CONTINUE
   DO 40 I=1,ND
   READ (I5,*) IDF(I)
40 CONTINUE
   DO 45 I=1,NS
   READ (I5,*) ISF(I)
45 CONTINUE
   DO 50 I=1,NSF
   READ (I5,*) IRESF(I)
50 CONTINUE
   DO 55 I=1,NSF
   READ (I5,*) ISFF(I)
55 CONTINUE
   DO 60 I=1,2
   DO 60 J=1,NSF
   READ (I5,*) RESF(I,J)
60 CONTINUE
   DO 65 I=1,NF
   DO 65 J=1,(N+NH+NT)*3
   READ (I5,*) FTSC(I,J)
65 CONTINUE
   DO 70 I=1, NF
   READ (I5,*) RHSF(I)
70 CONTINUE
   D0 75 I=1, NH
   READ (I5,*) RHSP2(I)
75 CONTINUE
   D0 80 I=1,NH*6
   READ (I5,*) PWSC(I)
80 CONTINUE
   DO 85 I=1,NH*6
```

```
      READ (I5,*) RESC(I,J)
   85 CONTINUE
      DO 90 I=1,NS
      READ (I5,*) OBFS(I)
   90 CONTINUE
      DO 95 I=1,NS
      READ (I5,*) RHSP3(I)
   95 CONTINUE
      DO 100 I=1,NS*6
      READ (I5,*) RESCS(I)
  100 CONTINUE
      DO 105 I=1,NS*6
      READ (I5,*) PWSCS(I)
  105 CONTINUE
      DO 110 I=1,NH*6+NT*6
      READ (I5,*) OBSC(I)
  110 CONTINUE
      DO 115 I=1,M
      DO 115 J=1,NE
      READ (I5,*) ISDRE(I,J)
  115 CONTINUE
      DO 120 I=1,M
      DO 120 J=1,ND
      READ (I5,*) ISDDF(I,J)
  120 CONTINUE
      DO 125 I=1,M
      DO 125 J=1,NH
      READ (I5,*)  ISDFF(I,J)
  125 CONTINUE
      DO 130 I=1,M
      DO 130 J=1,NS
      READ (I5,*) ISDFS(I,J)
  130 CONTINUE
      DO 135 I=1, M
      DO 135 J=1,(N+NH+NT)*3
      READ (I5,*) SDSC(I,J)
  135 CONTINUE
      DO 140 I=1,M
      READ (I5,*) RHSS(I)
  140 CONTINUE
      DO 145 I=1,M
      DO 145 J=1,B
      READ (I5,*) ICYC(I,J)
  145 CONTINUE
      CLOSE (I5)

C INITIALIZE THE ROWS
      CALL INIT
C     GENERATE THE ROWS
```

```
C     OBJECTIVE FUNCTION
C     MIN SUM(J: Q(J)HF(J) + q(J)HF(J))
      CALL DEFROW(1,0.,IDROW,TRUBLE)
C     POWER CONSTRAINTS
      DO 200 I=1,NPW
      CALL DEFROW(-1,RHSP(I),IDROW,TRUBLE)
  200 CONTINUE
C     FIRST LAW CONSTRAINTS
      DO 210 I=1,NF
      CALL DEFROW(0,RHSF(I),IDROW,TRUBLE)
  210 CONTINUE
C     SECOND LAW CONSTRAINTS
      DO 220 I=1, M
      CALL DEFROW(0,RHSS(I),IDROW,TRUBLE)
  220 CONTINUE
C     CONVEX SET CONSTRAINTS
      DO 230 I=1,N
      CALL DEFROW(0,1,IDROW,TRUBLE)
  230 CONTINUE
C     REFERENCE CONSTRAINTS
      DO 240 I=1,NH+NS
      CALL (0,0,IDROW,TRUBLE)
  240 CONTINUE
C
      DO 250 I=1,N
      ISET1(I)=0
      IF (ISET3(I).GT.0.AND.ISET3(I).GT.1000) THEN
      K=NBACK(NS,ISF,ISET3(I))
      IF (K.NE.0)
      ISET1(I)=1
      ELSE
      IF (ISET3(I).GT.1000.AND.ISET3(I).LT.1000000) THEN
      J=ISET3(I)/1000
      K=NBACK(NS,ISF,J)
      IF (K.NE.0)
      ISET1(I)=1
      K=ISET3(I)-J*1000
      K=NBACK(NS,IDF,K)
      IF (K.NE.0)
      ISET1(I)=ISET1(I)+1
      ELSE
      IF (ISET3(I).GT.1000000) THEN
      J=ISET3(I)/1000000
      K=NBACK(NS,ISF,J)
      IF (K.NE.0)
      SET1(I)=1
      J1=(ISET3(I)-J*1000000)/1000
      K=NBACK(NS,ISF,J1)
      IF (K.NE.0)
      ISET1(I)=ISET1(I)+1
      K=ISET3(I)-J*1000000-J1*1000
      K=NBACK(NS,ISF,K)
      IF (K.NE.0)
      ISET1(I)=ISET1(I)+1
      ENDIF
      ENDIF
  250 CONTINUE
C
C GENERATE SET VARIABLES(IN OBJ.,POWER,1ST,2ND,CONVEX,REF.)
C
       L=1
      DO 300 I=1,NH
      VNAME(1)=ALFANM(34)
      VNAME(6)=BLANK
      VNAME(7)=BLANK
      VNAME(8)=BLANK
      CALL NAME (ISET(I),VNAME(2),VNAME(3),VNAME(4))
      DO 300 K=1,6
      NONZ=1
      IRO(1)=1
      VNAME(5)=K
      VAL(1)=OBSC((I-1)*6+K)
      NONZ=2
      IRO(2)=1+I+ND
      VAL(2)=PWSC((I-1)*6+K)
      IF (ISET1(I).EQ.0) GOTO 320
      DO 315 J=1, ISET1(I)
      NONZ=NONZ+1
      IRO(NONZ)=1+NH+ND+L
      VAL(NONZ)=PWSCS((L-1)*6+K)
      L=L+1
  315 CONTINUE
      L=L-ISET1(I)
  320 DO 330 J=1,NF
      IF (FTSC(J,(I-1)*6+K).EQ.0) GOTO 330
      NONZ=NONZ+1
      IRO(NONZ)=1+NPW+J
      VAL(NONZ)=FTSC(J,(I-1)*6+K)
  330 CONTINUE
      DO 340 J=1,M
      IF (SDSC(J,(I-1)*6+K).EQ.0) GOTO 340
      NONZ=NONZ+1
      IRO(NONZ)=1+NPW+NF+J
      VAL(NONZ)=SDSC(J,(I-1)*6+K)
  340 CONTINUE
      NONZ=NONZ+1
      IRO(NONZ)=1+NPW+NF+M+I
      VAL(NONZ)=1
      NONZ=NONZ+1
      IRO(NONZ)=1+NPW+NF+M+N+I
      VAL(NONZ)=RESC((I-1)*6+K)
      IF (ISET1(I).EQ.0) GOTO 350
      DO 345 J=1 ISET1(I)
      NONZ=NONZ+1
      IRO(NONZ)=1+NPW+NF+M+NH+ND+L
      VAL(NONZ)=RESCS((L-1)*6+K)
      L=L+1
  345 CONTINUE
  350 CALL APPCOL (VNAME,NONZ,VAL,IRO,TRUBLE)
  300 CONTINUE
C
C  GENERATE SET VARIABLE (NST:IN 1ST,2ND,CONVEX,OBJ,PW,REF)
C
      DO 360 I=NH+1,NH+NT
      VNAME(1)=ALFANM(34)
      CALL NAME (ISET(I),VNAME(2),VNAME(3),VNAME(4))
      VNAME(6)=BLANK
      VNAME(7)=BLANK
      VNAME(8)=BLANK
      DO 360 J=1, 6
      NONZ=1
      IRO(1)=1
      VNAME(5)=J
      VAL(1)=OBSC((I-1)*6+J)
      DO 375 K=1,ISET1(I)
      NONZ=NONZ+1
      IRO(NONZ)=1+NH+ND+L
      VAL(NONZ)=PWSCS((L-1)*6+J)
      L=L+1
      K=K+1
  375 CONTINUE
      L=L-ISET1(I)
      DO 380 K=1,NF
      IF (FTSC(K,(I-1)*6+J).EQ.0) GOTO 380
      NONZ=NONZ+1
      IRO(NONZ)=1+NPW+K
      VAL(NONZ)=FTSC(K,(I-1)*6+J)
  380 CONTINUE
      DO 385 K=1,M
      IF(SDSC(K,(I-1)*6+J).EQ.0) GOTO 385
      NONZ=NONZ+1
      IRO(NONZ)=1+NPW+NF+K
      VAL(NONZ)=SDSC(K,(I-1)*6+J)
  385 CONTINUE
      NONZ=NONZ+1
      IRO(NONZ)=1+NPW+NF+M+I
      VAL(NONZ)=1
      DO 390 K=1, ISET1(I)
      NONZ=NONZ+1
      IRO(NONZ)=1+NPW+NF+M+NH+ND+L
      VAL(NONZ)=RESCS((L-1)*6+J)
      L=L+1
  390 CONTINUE
      CALL APPCOL(VNAME,NONZ,VAL,IRO,TRUBLE)
  360 CONTINUE
C
C  GENERATE SET VARIABLE (NSET-NQH-NST:IN 1ST,2ND,CONVEX)
C
      DO 410 I=NH+NT+1,N
      VNAME(1)=ALFANM(34)
      CALL NAME (ISET(I),VNAME(2),VNAME(3),VNAME(4))
      VNAME(6)=BLANK
      VNAME(7)=BLANK
      VNAME(8)=BLANK
      DO 410 J=1, 3
      VNAME(5)=J
      NONZ=0
      DO 420 K=1,NFT
      IF (FTSC(K,(I+NH+NT)*3+1).EQ.0) GOTO 420
      NONZ=NONZ+1
      IRO(NONZ)=1+NPW+K
      VAL(NONZ)=FTSC(K,(I+NH+NT-1)*3+J)
  420 CONTINUE
      DO 425 K=1,M
      IF(SDSC(K,(I+NH+NT)*3+1).EQ.0) GOTO 425
      NONZ=NONZ+1
      IRO(NONZ)=1+NPW+NF+K
      VAL(NONZ)=SDSC(K,(I+NH+NT-1)*3+J)
  425 CONTINUE
      NONZ=NONZ+1
      IRO(NONZ)=1+NPW+NF+M+I
      VAL(NONZ)=1
      CALL APPCOL(VNAME,NONZ,VAL,IRO,TRUBLE)
  410 CONTINUE
C
C   GENERATE HR VARIABLES(IN SECOND LAW CONSTRAINTS)
C
      DO 430 I=1,NE
      VNAME(1)=ALFANM(18)
      VNAME(2)=ALFANM(28)
      VNAME(6)=BLANK
      VNAME(7)=BLANK
      VNAME(8)=BLANK
      CALL NAME (IRE(I),VNAME(3),VNAME(4),VNAME(5))
      NONZ=0
      DO 435 J=1,M
      IF (ISDRE(J,I).EQ.0) GOTO 435
      NONZ=NONZ+1
      IRO(NONZ)=1+NPW+NF+J
      VAL(NONZ)=ISDRE(J,I)
  435 CONTINUE
      CALL APPCOL(VNAME,NONZ,VAL,IRO,TRUBLE)
  430 CONTINUE
C
C    GENRATE HF VARIABLE IN FIXED BRANCHES(IN OBJ.,PW,2ND)
C
      DO 440 I=1,ND
      VNAME(1)=ALFANM(18)
      VNAME(2)=ALFANM(16)
      VNAME(6)=BLANK
      VNAME(7)=BLANK
      VNAME(8)=BLANK
      CALL NAME(IDF(I),VNAME(3),VNAME(4),VNAME(5))
      NONZ=1
      IRO(1)=1
      VAL(1)=OBDF(I)
      NONZ=2
      IRO(2)=1+I
      VAL(2)=1
      DO 445 J=1, M
      IF (ISDDF(J,I).EQ.0) GOTO 445
      NONZ=NONZ+1
      IRO(NONZ)=1+NPW+NF+J
      VAL(NONZ)=ISDDF(J,I)
  445 CONTINUE
      CALL APPCOL(VNAME,NONZ,VAL,IRO,TRUBLE)
  440 CONTINUE
C
C     GENERATE HF IN FREE BRANCHES (IN SECOND & REF.)
C
      DO 450 I=1,NH
      VNAME(1)=ALFANM(18)
      VNAME(2)=ALFANM(16)
      VNAME(6)=BLANK
      VNAME(7)=BLANK
      VNAME(8)=BLANK
      CALL NAME (ISET(I),VNAME(3),VNAME(4),VNAME(5))
      NONZ=0
      DO 455 J=1, M
      IF(ISDFF(J,I).EQ.0) GOTO 455
      NONZ=NONZ+1
      IRO(NONZ)=1+NPW+NF+J
      VAL(NONZ)=ISDFF(J,I)
  455 CONTINUE
      NONZ=NONZ+1
      IRO(NONZ)=1+NPW+NF+M+N+I
      VAL(NONZ)=-1
      CALL APPCOL(VNAME,NONZ,VAL,IRO,TRUBLE)
  450 CONTINUE
C
C    GENERATE HF OUT OF SET (IN OBJ.,POWER,2ND, REF.)
C
      DO 460 I=1,NS
      VNAME(1)=ALFANM(18)
      VNAME(2)=ALFANM(16)
      VNAME(6)=BLANK
      VNAME(7)=BLANK
      VNAME(8)=BLANK
      CALL NAME(ISF(I),VNAME(3),VNAME(4),VNAME(5))
      NONZ=1
      IRO(1)=1
      VAL(1)=OBFS(I)
      NONZ=2
      IRO(2)=1+ND+NH+I
      VAL(2)=OBFS(I)
      DO 465 J=1,M
      IF (ISDFS(J,I).EQ.0) GOTO 465
      NONZ=NONZ+1
      IRO(NONZ)= 1+NPW+NF+J
      VAL(NONZ)=ISDFS(J,I)
  465 CONTINUE
      NONZ=NONZ+1
      IRO(NONZ)=1+NPW+NF+M+N+NH+I
      VAL(NONZ)=-1
      CALL APPCOL(VNAME,NONZ,VAL,IRO,TRUBLE)
  460 CONTINUE
C
C     END OF MATRIX FORMATION
C
      LISTC=0
      SOLI=100000000
      SOL=SOLI
C
      SOLVE THE CURRENT PROBLEM
C
  500 CALL GO (0,ISTAT)
C
      IF (ISTAT.NE.2) GOTO 600
C
C     TAKE THE LAST PROBLEM FROM THE LIST
C
  510 IF (LISTC.GT.0) THEN
      IF (ICOUNT.EQ.LIST(LISTC)
      .AND.DLIST(LISTC).SET(3)-SET(1)) THEN
      TEMP=SET(1)
      SET(1)=SET(3)
      SET(2)=2SET(1)-SET(2)
      SET(3)=2SET(1)-TEMP
      CALL UPDATA(SETPTS,ICOUNT,SET)
      CALL GCAL (ICOUNT,SET)
      ELSE
      DO 520 I=1,NH
      CALL CSET(SET,LISTC,I,BS)
      CALL UPDATA (SETPTS,I,SET)
      CALL CNQH(I,SET)
  520 CONTINUE
      DO 530 I=NH+1,NH+NT
      CALL CSET(SET,LISTC,I,BS)
      CALL UPDATA (SETPTS,I,SET)
      CALL CNST(I,SET)
  530 CONTINUE
      DO 540 I=NH+NT+1, N
      CALL CSET(SET,LISTC,I,BS)
      CALL UPDATA (SETPTS,I,SET)
      CALL CNSET(I,SET)
  540 CONTINUE
      ENDIF
      ICOUNT=LIST(LISTC)
      DO 550 I=1,3
      SET(I)=SETPTS(ICOUNT,I)
  550 CONTINUE
      DO 560 I=1,N
      BS(LISTC,I,1)=0
      BS(LISTC,I,2)=0
  560 CONTINUE
      LIST(LISTC)=0
      DLIST(LISTC)=0
      LISTC=LISTC-1
      GOTO 500
      ENDIF
C
      IF (SOL.EQ.SOLI) THEN
      GOTO 2000
      ELSE
      GOTO 1900
      ENDIF
C     CHECK SOS CONDITION
C
  600 DISM=0.25
      NS3=0
      NS2=0
      DO 610 I=1,NH+NT
      J=1
      ISOL(I)=0
      TSOL(I)=0
  620 WHILE (J.LE.3)
      CALL REPVAR((I-1)*6+J,PRIMAL,DUAL)
      S1=PRIMAL
      IF (S1.NE.0)
      TSOL(I)=TSOL(I)+S1*SETPTS(I,J)
      CALL REPVAR((I-1)*6+J+3,PRIMAL,DUAL)
      S2=PRIMAL
      IF (S2.NE.0)
      TSOL(I)=TSOL(I)+S2*SETPTS(I,J)
      IF (S1.EQ.0.AND.S2.EQ.0) THEN
      INDEX(J)=0
      J=J+1
      GOTO 620
      ENDIF
      INDEX(J)=J
      ISOL(I)=ISOL(I)+1
      J=J+1
      ENDWHILE
      IF (ISOL(I).EQ.2.AND.INDEX(2).EQ.0)
      ISOL(I)=3
      IF (ISOL(I).EQ.2)
      NS2=NS2+1
      IF(ISOL(I).EQ.3) THEN
      NS3=NS3+1
      DIS=SETPTS(I,3)-SETPTS(I,1)
      IF (DIS.GT.DISM) THEN
      DISM=DIS
      ICT=I
      ENDIF
      ENDIF
  610 CONTINUE
C
      DO 630 I=NH+NT+1,N
      J=1
      ISOL(I)=0
      TSOL(I)=0
  640 WHILE (J.LE.3)
      CALL REPVAR((I+NH+NT-1)*3+J,PRIMAL,DUAL)
      IF (PRIMAL.NE.0) THEN
      TSOL(I)=TSOL(I)+PRIMAL*SETPTS(I,J)
      ELSE
      INDEX(J)=0
      J=J+1
      GOTO 640
      ENDIF
      INDEX(J)=J
      ISOL(I)=ISOL(I)+1
      J=J+1
      ENDWHILE
      IF (ISOL(I).EQ.2.AND.INDEX(2).EQ.0)
      ISOL(I)=3
      IF (ISOL(I).EQ.2)
      NS2=NS2+1
      IF(ISOL(I).EQ.3) THEN
      NS3=NS3+1
      DIS=SETPTS(I,3)-SETPTS(I,1)
      IF (DIS.GT.DISM) THEN
      DISM=DIS
      ICT=I
      ENDIF
      ENDIF
  630 CONTINUE
C
      NCOND=1
      DO 650 I=1,N
      IF(ISOL(I).GT.NCOND)
      NCOND=ISOL(I)
  650 CONTINUE
C
      GOTO(700,800,900) NCOND
C
  700 CALL REPROW(1,PRIMAL,DUAL)
      IF (PRIMAL.LT.SOL) THEN
      SOL=PRIMAL
C     SAVE THE SOLUTION
      CALL SDBC (I5)
      DO 710 I=1,N
      DO 710 J=1,3
      SAVSTS(I,J)=SETPTS(I,J)
  710 CONTINUE
      ENDIF
      GOTO 510
C
C     CHECK THE CONVERGENCE
C
  800 DO 810 I=1,M
      TEST=0
      DO 820 J=1,B
      IF (ABS(ICYC(I,J)).NE.0) THEN
      ISIGN=ICYC(I,J)/ABS(ICYC(I,J))
      IF (ABS(ICYC(I,J))/100.EQ.1) THEN
      L=0
      K=NBACK(NH,ISET,J)
      IF (K.EQ.0) THEN
      L=1
      K=NBACK(NS,ISF,J)
      ENDIF
      IF (K.EQ.0) THEN
      L=2
      K=NBACK(ND,IDF,J)
      ENDIF
      IF (L.EQ.0) THEN
      K=(N+NH+NT)*3+ND+NE+K
      ELSE
      IF (L.EQ.1) THEN
      K=(N+NH+NT)*3+ND+NE+NH+K
      ELSE
      K=(N+NH+NT)*3+NE+K
      ENDIF
      ENDIF
      CALL REPVAR(K,PRIMAL,DUAL)
      TEST=TEST-ISIGN*PRIMAL
      ENDIF
      IF (ABS(ICYC(I,J))/100.EQ.1) THEN
      ITEMP=ABS(ICYC(I,J))-100
      ELSE
      ITEMP=ABS(ICYC(I,J))
      ENDIF
      IF (ITEMP/10.EQ.1) THEN
      K=NBACK(NE,IRE,J)
      K=(N+NH+NT)*3+K
      CALL REPVAR(K,PRIMAL,DUAL)
      TEST=TEST+ISIGN*PRIMAL
      ENDIF
      IF(ITEMP/10.EQ.1)
      ITEMP=ITEMP-10
      IF (ITEMP.EQ.1) THEN
      K=NBACK(N,ISET,J)
      L=0
      IF (K.EQ.0) THEN
      K=NBACK(NSF,ISFF,J)
      L=1
      ENDIF
      IF (L.EQ.0) THEN
      TEST=TEST+ISIGN*REST(K)*ABS(TSOL(K))*TSOL(K)
      ELSE
      IS=IRESF(K)/ABS(IRESF(K))
      TEST=TEST+
      ISIGN*RESF(K,1)*ABS(IS*TSOL(ABS(IRESF(K)))
      +RESF(K,2))*(IS*TSOL(ABS(IRESF(K)))+RESF(K,2))
      ENDIF
      ENDIF
      ENDIF
  820 CONTINUE
      EXM(I)=ABS(TEST-RHSS(I))
  810 CONTINUE
C
      BIG=EXM(1)
      DO 830 I=2,M
      IF (EXM(I).GT.BIG)
      BIG=EXM(I)
  830 CONTINUE
      IF (BIG.LT.ER) GOTO 700
C
C     ADD THE MIDDLE POINT
C
      I=1
      WHILE (I.LE.N)
      IF (ISOL(I).EQ.2) THEN
      NS2=NS2-1
      ICOUNT=I
C
      IF (ICOUNT.LE.NH+NT) THEN
      K=(ICOUNT-1)*6
      J=1
  840 WHILE (J.LE.6)
      CALL REPVAR(K+J,PRIMAL,DUAL)
      IF (PRIMAL.EQ.0) THEN
      J=J+1
      GOTO 840
      ENDIF
      IFT=J
      J=J+1
      ENDWHILE
      IF (IFT.EQ.2.OR.IFT.EQ.5) THEN
      IFT=1
      ELSE
      IFT=2
      ENDIF
      ELSE
      K=(ICOUNT+NH+NT-1)*3
      IFT=1
      CALL REPVAR (K+1,PRIMAL,DUAL)
      IF (PRIMAL.EQ.0)
      IFT=2
      ENDIF
      IF (IFT.EQ.1) THEN
      SET(1)=SETPTS(ICOUNT,1)
      SET(3)=SETPTS(ICOUNT,2)
      SET(2)=(SET(1)+SET(3))/2
      ELSE
      SET(1)=SETPTS(ICOUNT,2)
      SET(3)=SETPTS(ICOUNT,3)
      SET(2)=(SET(1)+SET(3))/2
      ENDIF
      CALL GCAL(ICOUNT,SET)
      CALL GO (0,ISTAT)
      IF (ISTAT.EQ.2) THEN
      IF (NS2.EQ.0) THEN
      CALL UPDATA (SETPTS,ICOUNT,SET)
      GOTO 510
      ELSE
      GOTO 845
      ENDIF
      ENDIF
      CALL REPROW(1,PRIMAL,DUAL)
      IF (PRIMAL,GT.SOL) THEN
      IF (NS2.EQ.0) THEN
      CALL UPDATA (SETPTS,ICOUNT,SET)
      GOTO 510
      ELSE
      GOTO 845
      ENDIF
      ELSE
      CALL UPDATA(SETPTS,ICOUNT,SET)
      GOTO 600
  845 IF (IFT.EQ.1) THEN
      SET(1)=SET(1)
      SET(2)=SET(3)
      SET(3)=2SET(2)-SET(1)
      ELSE
      SET(2)=SET(1)
      SET(3)=SET(3)
      SET(1)=2SET(2)-SET(3)
      ENDIF
      CALL GCAL(ICOUNT,SET)
      ENDIF
      I=I+1
      ENDWHILE
C
C     NEED TO DIVIDE THE SETS
C
  900 NCOND=ICT
      DISM=0.25
      IF(NS3.GT.1.AND.ICT.GT.NH) THEN
      DO 905 I=1,NH
      IF (ISOL(I).EQ.3) THEN
      DIS=SETPTS(I,3)-SETPTS(I,1)
      IF (DIS.GT.DISM) THEN
      DISM=DIS
      NCOND=I
      ENDIF
      ENDIF
  905 CONTINUE
      ENDIF
      ICOUNT=NCOND
C
C     BRANCH THE PROBLEM INTO TWO SUBPROBLEMS
C
      SET(1)=SETPTS(ICOUNT,2)
      SET(3)=SETPTS(ICOUNT,3)
      SET(2)=(SET(1)+SET(3))/2
      CALL UPDATA(SETPTS,ICOUNT,SET)
      LISTC=LISTC+1
      LIST(LISTC)=ICOUNT
      DLIST(LISTC)=SET(3)-SET(1)
      DO 910 I=1,N
      BS(LISTC,I,1)=SETPTS(I,1)
      BS(LISTC,I,2)=SETPTS(I,3)
  910 CONTINUE
      TEMP=SET(3)
      SET(3)=SET(1)
      SET(2)=2SET(1)-SET(2)
      SET(1)=2SET(1)-TEMP
      CALL UPDATA(SETPTS,ICOUNT,SET)
      CALL GCAL(ICOUNT,SET)
      GOTO 500
C     THE CURRENT SOLUTION IS OPTIMAL
 1900 RDBC (I5)
      DO 1910 I=1, NH+NT
      TSOL(I)=0
      DO 1920 J=1,3
      CALL REPVAR((I-1)*6+J,PRIMAL,DUAL)
      TSOL(I)=TSOL(I)+SAVSTS(I,J)*PRIMAL
      CALL REPVAR((I-1)*6+J+3,PRIMAL,DUAL)
      TSOL(I)=TSOL(I)+SAVSTS(I,J)
 1920 CONTINUE
      IF (ISET3(I).GT.0.AND.ISET3(I).LT.1000) THEN
      K=NBACK(NSF,ISFF,ISET3(I))
      REPS(I)(K)=TSOL(I)+RESF(K,2)
      ELSE
      IF (ISET3(I).GT.1000.AND.ISET3(I).LT.1000000)
      K=ISET3(I)/1000
      L=NBACK(NSF,ISFF,K)
      REPS(L)=TSOL(I)+RESF(L,2)
      K=ISET3(I)-K*1000
      K=NBACK(NSF,ISFF,K)
      REPS(K)=TSOL(I)+RESF(K,2)
      ELSE
      IF (ISET3(I).GT.1000000) THEN
      K=ISET3(I)/1000000
      L=NBACK(NSF,ISFF,K)
      REPS(L)=TSOL(I)+RESF(L,2)
      L=(ISET3(I)-K*1000000)/1000
      L1=NBACK(NSF,ISFF,L)
      REPS(L1)=TSOL(I)+RESF(L1,2)
      K=ISET3(I)-K*1000000-L*1000
      K=NBACK(NSF,ISFF,K)
      REPS(K)=TSOL(I)+RESF(K,2)
      ENDIF
      ENDIF
      ENDIF
 1910 CONTINUE
      DO 1930 I=NH+NT,N
      TSOL(I)=0
      DO 1940 J=1,3
      CALL REPVAR((I-1)*6+J,PRIMAL,DUAL)
      TSOL(I)=TSOL(I)+SAVSTS(I,J)*PRIMAL
 1940 CONTINUE
      IF (ISET3(I).GT.0.AND.ISET3(I).LT.1000) THEN
      K=NBACK(NSF,ISFF,ISET3(I))
      REPS(I)(K)=TSOL(I)+RESF(K,2)
      ELSE
      IF (ISET3(I).GT.1000.AND.ISET3(I).LT.1000000) THEN
      K=ISET3(I)/1000
      L=NBACK(NSF,ISFF,K)
      REPS(L)=TSOL(I)+RESF(L,2)
      K=ISET3(I)-K*1000
      K=NBACK(NSF,ISFF,K)
      REPS(K)=TSOL(I)+RESF(K,2)
      ELSE
      IF (ISET3(I).GT.1000000) THEN
      K=ISET3(I)/1000000
      L=NBACK(NSF,ISFF,K)
      REPS(L)=TSOL(I)+RESF(L,2)
      L=(ISET3(I)-K*1000000)/1000
      L1=NBACK(NSF,ISFF,L)
      REPS(L1)=TSOL(I)+RESF(L1,2)
      K=ISET3(I)-K*1000000-L*1000
      K=NBACK(NSF,ISFF,K)
      REPS(K)=TSOL(I)+RESF(K,2)
      ENDIF
      ENDIF
      ENDIF
 1930 CONTINUE
      CALL OUTSPC (1)
      DO 1950 I=1,N
      CA(I)(1)=ALFANM(27)
      CALL NAME (ISET(I), CA(I)(2),CA(I)(3),CA(I)(4))
 1950 CONTINUE
      DO 1960 I=1,NSF
      CC(I)(1)=ALFANM(27)
      CALL NAME (ISFF(I), CC(I)(2),CC(I)(3),CC(I)(4))
 1960 CONTINUE
      DO 1970 I=1,N
      WRITE (I6,1990) CA(I), TSOL(I)
 1970 CONTINUE
      DO 1980 I=1,NSF
      WRITE (I6,1990) CC(I), REPS(I)
 1980 CONTINUE
      GOTO 2010
C   NO FEASIBLE SOLUTION, CHECK THE PROBLEM FORMATION
 2000 WRITE (*,'(//,5X,''NO FEASIBLE SOLUTION'',
     *,''CHECK THE PROBLEM FORMATION'')')
 2010 STOP
      END
 1990 FORMAT (1H ,1X,A4,1X,1H=,1X,F7.3)
C
      SUBROUTINE NAME(INDEX,K1,K2,K3)
      CHARACTER K1,K2,K3
      IF (INDEX.LT.10) THEN
      K1='INDEX'
      K2='0'
      K3='0'
      ELSE
      IF (INDEX.LT.100) THEN
      K1='INDEX/10'
      K2='INDEX-K1*10'
      K3='0'
      ELSE
      K1='INDEX/100'
      K2='(INDEX-K1*100)/10'
      K3='INDEX-K1*100-K2*10'
      ENDIF
      ENDIF
      RETURN
      END
C
      FUNCTION NABCK(NUM,INDEX,J)
      NBACK=0
      DO 10 I=1,NUM
      IF (J.EQ.ABS(INDEX(I))) THEN
      NBACK=I
      GOTO 20
      ENDIF
   10 CONTINUE
   20 RETURN
      END
C
      SUBROUTINE  UPDATA (SET1,I,SET)
      DO 10 J=1,3
      SET1(I,J)=SET(J)
   10 CONTINUE
      RETURN
      END
C
      SUBROUTINE GCAL (N1,SET)
      COMMON /USERCM/NH,NT,NPW,NF,M,ISET(N),REST(N,3)
     1 FTSC(NF,(N+NH+NT)*3),SDSC(M,(N+NH+NT)*3)
      IF (N1,GT.NH+NT) THEN
      CALL CNSET(N1,SET)
      ELSE
      IF (N1.LE.NH) THEN
      CALL CNQH(N1,SET)
      ELSE
      CALL CNST(N1,SET)
      ENDIF
      ENDIF
      RETURN
      END
C
      SUBROUTINE CSET(SET,N,M,BS)
      SET(1)=BS(N,M,1)
      SET(3)=BS(N,M,2)
      SET(2)=(SET(1)+SET(3))/2
      RETURN
      END
C
```

```
      SUBROUTINE CNSET (N1,SET)
      COMMON /USERCM/ NH,NT,NPW,NF,M,ND,NS,NSF
     1 ISF(NS),RESC(NH*6),RESCS(NS*6),REST(N)
     2 FTSC(NF,(N+NH+NT)*3),SDSC(M,(N+NH+NT)*3)
     3 ISET3(N),ISFF(NSF),IRESF(NSF),RESF(NSF,2)
      DO 10 I=1,3
      K=(N+NH+NT-1)*3
      DO 20 J=1,NFT
      IF (FTSC(J,K+1).NE.0)
      CALL INSERT(1+NPW+J,K+I,SET(I),1)
   20 CONTINUE
      DO 30 J=1,M
      IF (SDSC(J,K+1).NE.0) THEN
      AMOUNT=REST(N)*ABS(SET(I))*SET(I)
      IF (ISET3(N1).GT.0.AND.ISET3(N1).LT.1000) THEN
      L=NBACK(NSF,ISFF,ISET3(N1))
      IS=IRESF(L)/ABS(IRESF(L))
      AMOUNT=AMOUNT+RESF(L,1)*ABS(IS*SET(I)+RESF(L,2))
             *(IS*SET(I)+RESF(L,2))
      ELSE
      IF(ISET3(N1).GT.1000.AND.ISET3(N1).LT.1000000) THEN
      L=ISET3(N1)/1000
      L1=NBACK(NFF,ISFF,L)
      IS=IRESF(L1)/ABS(IRESF(L1))
      AMOUNT=AMOUNT+RESF(L1,1)*ABS(IS*SET(I)+RESF(L1,2))
             *(IS*SET(I)+RESF(L1,2))
      L=ISET3(N1)-L*1000
      L=NBACK(NSF,ISFF,L)
      IS=IRESF(L)/ABS(IRESF(L))
      AMOUNT=AMOUNT+RESF(L,1)*ABS(IS*SET(I)+RESF(L,2))
             *(IS*SET(I)+RESF(L,2))
      ELSE
      IF (ISET3(N1).GT.1000000) THEN
      L=ISET3(N1)/1000000
      L1=NBACK(NFF,ISFF,L)
      IS=IRESF(L1)/ABS(IRESF(L1))
      AMOUNT=AMOUNT+RESF(L1,1)*ABS(IS*SET(I)+RESF(L1,2))
             *(IS*SET(I)+RESF(L1,2))
      L1=(ISET3(N1)-L*1000000)/1000
      L2=NBACK(ISF,ISFF,L1)
      IS=IRESF(L2)/ABS(IRESF(L2))
      AMOUNT=AMOUNT+RESF(L2,1)*ABS(IS*SET(I)+RESF(L2,2))
             *(IS*SET(I)+RESF(L2,2))
      L=ISET3(N1)-L*1000000-L1*1000
      L=NBACK(NSF,ISFF,L)
      IS=IRESF(L)/ABS(IRESF(L))
      AMOUNT=AMOUNT+RESF(L,1)*ABS(IS*SET(I)+RESF(L,2))
```

```
              *(IS*SET(I)+RESF(L,2))
      ENDIF
      ENDIF
      ENDIF
      CALL INSERT(1+NPW+NF+J,K+I,AMOUNT,1)
      ENDIF
   30 CONTINUE
   10 CONTINUE
      RETURN
      END
C
      SUBROUTINE CNST (N,SET)
      COMMON /USERCM/ NH,NT,NPW,NF,M,ND,NS,NSF
     1 ISF(NS),RESC(NH*6),RESCS(NS*6),REST(N)
     2 FTSC(NF,(N+NH+NT)*3),SDSC(M,(N+NH+NT)*3)
     3 ISET3(N),ISFF(NSF),IRESF(NSF),RESF(NSF,2)
      DO 10 I=1,3
      AMOUNT=REST(N1)*ABS(SET(I))*SET(I)
      FL=0
      FU=0
      IF (ISET3(N1).GT.0.AND.ISET3(N1).LT.1000) THEN
      K=NBACK(NS,ISF,ISET3(N1))
      FL=RESCS((K-1)*6+1)
      FU=RESCS((K-1)*6+4)
      CALL INSERT (1+NH+ND+K,(N-1)*6+I,FL*SET(I),1)
      CALL INSERT (1+NH+ND+K,(N-1)*6+I+3,FU*SET(I),1)
      L=NBACK(NSF,ISFF,ISET3(N1))
      IS=IRESF(L)/ABS(IRESF(L))
      AMOUNT=AMOUNT+RESF(L,1)*ABS(IS*SET(I)+RESF(L,2))
              *(IS*SET(I)+RESF(L,2))
      ELSE
      IF(ISET3(N1).GT.1000.AND.ISET3(N1).LT.1000000)THEN
      L=ISET3(N1)/1000
      L1=NBACK(NSF,ISFF,L)
      IS=IRESF(L1)/ABS(IRESF(L1))
      AMOUNT=AMOUNT+RESF(L1,1)*ABS(IS*SET(I)+RESF(L1,2))
      *(IS*SET(I)+RESF(L1,2))
      K=NBACK(NS,ISF,L)
      IF (K.NE.0) THEN
      FL=RESCS((k-1)*6+1)
      FU=RESCS((K-1)*6+4)
      CALL INSERT (1+NH+ND+K,(N-1)*6+I,FL*SET(I),1)
      CALL INSERT (1+NH+ND+K,(N-1)*6+I+3,FU*SET(I),1)
      ENDIF
      L=ISET3(N1)-L*1000
      K=NBACK(NS,ISF,L)
      IF (K.NE.0) THEN
      FL1=RESCS((K-1)*6+1)
      FU1=RESCS((K-1)*6+4)
```

```
CALL INSERT (1+NH+ND+K,(N-1)*6+I,FL1*SET(I),1)
CALL INSERT (1+NH+ND+K,(N-1)*6+I+3,FU1*SET(I),1)
FL=FL+FL1
FU=FU+FU1
ENDIF
L=NBACK(NSF,ISFF,L)
IS=IRESF(L)/ABS(IRESF(L))
AMOUNT=AMOUNT+RESF(L,1)*ABS(IS*SET(I)+RESF(L,2))
       *(IS*SET(I)+RESF(L,2))
ELSE
L=ISET3(N1)/1000000
L1=NBACK(NSF,ISFF,L)
K=NBACK(NS,ISF,L)
IS=IRESF(L1)/ABS(IRESF(L1))
AMOUNT=AMOUNT+RESF(L1,1)*ABS(IS*SET(I)+RESF(L1,2))
        *(IS*SET(I)+RESF(L1,2))
IF (K.NE.0) THEN
FL=RESCS((K-1)*6+1)
FU=RESCS((K-1)*6+4)
CALL INSERT (1+NH+ND+K,(N-1)*6+I,FL*SET(I),1)
CALL INSERT (1+NH+ND+K,(N-1)*6+I+3,FU*SET(I),1)
ENDIF
L1=(ISET3(N1)-L*1000000)/1000
L2=NBACK(NSF,ISFF,L1)
K=NBACK(NS,ISF,L1)
IS=IRESF(L2)/ABS(IRESF(L2))
AMOUNT=AMOUNT+RESF(L2,1)*ABS(IS*SET(I)+RESF(L2,2))
       *(IS*SET(I)+RESF(L2,2))
IF (K.NE.0) THEN
FL1=RESCS((K-1)*6+1)
FU1=RESCS((K-1)*6+4)
CALL INSERT (1+NH+ND+K,(N-1)*6+I,FL1*SET(I),1)
CALL INSERT (1+NH+ND+K,(N-1)*6+I+3,FU1*SET(I),1)
FL=FL+FL1
FU=FU+FU1
ENDIF
L=ISET3(N1)-L*1000000-L1*1000
K=NBACK(NS,ISF,L)
L=NBACK(NSF,ISFF,L)
IS=IRESF(L)/ABS(IRESF(L))
AMOUNT=AMOUNT+RESF(L,1)*ABS(IS*SET(I)+RESF(L,2))
       *(IS*SET(I)+RESF(L,2))
IF (K.NE.0) THEN
FL1=RESCS((K-1)*6+1)
FU1=RESCS((K-1)*6+4)
CALL INSERT (1+NH+ND+K,(N-1)*6+I,FL1*SET(I),1)
CALL INSERT (1+NH+ND+K,(N-1)*6+I+3,FU1*SET(I),1)
FL=FL+FL1
FU=FL+FU1
```

```
      ENDIF
      ENDIF
      ENDIF
      CALL INSERT (1,(N1-1)*6+I,FL*SET(I),1)
      CALL INSERT (1,(N1-1)*6+I+3,FU*SET(I),1)
      DO 20 J=1,NFT
      IF (FTSC(J,N*6).NE.0) THEN
      CALL INSERT(1+NPW+J,(N1-1)*6+I,SET(I),1)
      CALL INSERT(1+NPW+J,(N1-1)*6+I+3,SET(I),1)
      ENDIF
   20 CONTINUE
      DO 30 J=1,M
      IF (SDSC(J,N*6).NE.0) THEN
      CALL INSERT(1+NPW+NF+J,(N-1)*6+I,AMOUNT,1)
      CALL INSERT(1+NPW+NF+J,(N-1)*6+I+3,AMOUNT,1)
      ENDIF
   30 CONTINUE
   10 CONTINUE
      RETURN
      END
C
      SUBROUTINE CNQH (N1,SET)
      COMMON /USERCM/ NH,NT,NPW,NF,M,ND,NS,NSF
     1 ISF(NS),RESC(NH*6),RESCS(NS*6),REST(N)
     2 FTSC(NF,(N+NH+NT)*3),SDSC(M,(N+NH+NT)*3)
     3 ISET3(N),ISFF(NSF),IRESF(NSF),RESF(NSF,2)
      DO 10 I=1,3
      FL=RESC((N1-1)*6+1)
      FU=RESC((N1-1)*6+4)
      CALL INSERT (1+ND+N1, (N1-1)*6+I, FL*SET(I),1)
      CALL INSERT (1+ND+N1,(N1-1)*6+I+3,FU*SET(I),1)
      AMOUNT=REST(N)*ABS(SET(I))*SET(I))
      IF (ISET3(N1).GT.0.AND.ISET3(N1).LT.1000) THEN
      K=NBACK(NS,ISF,ISET3(N1))
      IF (K.NE.0) THEN
      FL1=RESCS((K-1)*6+1)
      FU1=RESCS((K-1)*6+4)
      CALL INSERT (1+ND+NH+K, (N1-1)*6+I, FL1*SET(I),1)
      CALL INSERT (1+ND+NH+K,(N1-1)*6+I+3,FU1*SET(I),1)
      FL=FL+FL1
      FU=FU+FU1
      ENDIF
      K=NBACK(NSF,ISFF,ISET3(N1))
      IS=IRESF(K)/ABS(IRESF(K))
      AMOUNT=AMOUNT+RESF(K,1)*ABS(IS*SET(I)+RESF(K,2))
             *(IS*SET(I)+RESF(K,2))
      ELSE
      IF(ISET3(N1).GT.1000.AND.ISET3(N1).LT.1000000)THEN
      L=ISET3(N1)/1000
```

```
K=NBACK(NS,ISF,L)
IF (K.NE.0) THEN
FL1=RESCS((K-1)*6+1)
FU1=RESCS((K-1)*6+4)
FL=FL+FL1
FU=FU+FU1
CALL INSERT (1+ND+NH+K,(N1-1)*6+I, FL1*SET(I),1)
CALL INSERT (1+ND+NH+K,(N1-1)*6+I+3,FL1*SET(I),1)
ENDIF
K=NBACK(NSF,ISFF,L)
IS=IRESF(K)/ABS(IRESF(K))
AMOUNT=AMOUNT+RESF(K,1)*ABS(IS*SET(I)+RESF(K,2))
       *(IS*SET(I)+RESF(K,2))
L=ISET3(N1)-L*1000
K=NBACK(NS,ISF,L)
IF (K.NE.0) THEN
FL1=RESCS((K-1)*6+1)
FU1=RESCS((K-1)*6+4)
CALL INSERT (1+ND+NH+K, (N1-1)*6+I, FL1*SET(I),1)
CALL INSERT (1+ND+NH+K,(N1-1)*6+I+3,FL1*SET(I),1)
FL=FL+FL1
FU=FU+FU1
ENDIF
K=NBACK(NSF,ISFF,L)
IS=IRESF(K)/ABS(IRESF(K))
AMOUNT=AMOUNT+RESF(K,1)*ABS(IS*SET(I)+RESF(K,2))
       *(IS*SET(I)+RESF(K,2))
ELSE
IF (ISET3(N1).GT.1000000) THEN
L=ISET3(N1)/1000000
K=NBACK(NSF,ISFF,L)
L1=NBACK(NS,ISF,L)
IS=IRESF(K)/ABS(IRESF(K))
AMOUNT=AMOUNT+RESF(K,1)*ABS(IS*SET(I)+RESF(K,2))
       *(IS*SET(I)+RESF(K,2))
IF (L1.NE.0) THEN
FL1=RESCS((L1-1)*6+1)
FU1=RESCS((L1-1)*6+4)
CALL INSERT (1+ND+NH+L1, (N1-1)*6+I, FL1*SET(I),1)
CALL INSERT (1+ND+NH+L1,(N1-1)*6+I+3,FU1*SET(I),1)
FU=FU+FU1
FL=FL+FL1
ENDIF
L1=(ISET3(N1)-L*1000000)/1000
K=NBACK(NSF,ISFF,L1)
L2=NBACK(NS,ISF,L1)
IS=IRESF(K)/ABS(IRESF(K))
AMOUNT=AMOUNT+RESF(K,1)*ABS(IS*SET(I)+RESF(K,2))
       *(IS*SET(I)+RESF(K,2))
```

```
   IF (L2.NE.0) THEN
   FL1=RESCS((L2-1)*6+1)
   FU1=RESCS((L2-1)*6+4)
   CALL INSERT (1+ND+NH+L2, (N1-1)*6+I, FL1*SET(I),1)
   CALL INSERT (1+ND+NH+L2,(N1-1)*6+I+3,FU1*SET(I),1)
   FL=FL+FL1
   FU=FU+FU1
   ENDIF
   L=ISET3(N1)-L*1000000-L1*1000
   K=NBACK(NSF,ISFF,L)
   L=NBACK(NS,ISF,L)
   IS=IRESF(K)/ABS(IRESF(K))
   AMOUNT=AMOUNT+RESF(K,1)*ABS(IS*SET(I)+RESF(K,2))
           *(IS*SET(I)+RESF(K,2))
   IF (L.NE.0) THEN
   FL1=RESCS((L-1)*6+1)
   FU1=RESCS((L-1)*6+4)
   CALL INSERT (1+ND+NH+L, (N1-1)*6+I, FL1*SET(I),1)
   CALL INSERT (1+ND+NH+L,(N1-1)*6+I+3,FU1*SET(I),1)
   FL=FL+FL1
   FU=FU+FU1
   ENDIF
   ENDIF
   ENDIF
   ENDIF
   CALL INSERT (1, (N1-1)*6+I, FL*SET(I),1)
   CALL INSERT (1, (N1-1)*6+I+3, FU*SET(I),1)
   DO 20 J=1,NF
   IF (FTSC(J,N1*6).NE.0) THEN
   CALL INSERT(1+NPW+J,(N1-1)*6+I,SET(I),1)
   CALL INSERT(1+NPW+J,(N1-1)*6+I+3,SET(I),1)
   ENDIF
20 CONTINUE
   DO 30 J=1,M
   IF (SDSC(J,N1*6).NE.0) THEN
   CALL INSERT(1+NPW+NF+J,(N1-1)*6+I,AMOUNT,1)
   CALL INSERT(1+NPW+NF+J,(N1-1)*6+I+3,AMOUNT,1)
   ENDIF
30 CONTINUE
10 CONTINUE
   RETURN
   END
```

VITA

Xing Wu was born in October 31, 1956 in Jixi, Heilongjiang, China. He graduated in July, 1979 with a B.S. degree in Mining Engineering from Heilongjiang Mining Institute. After that He worked for one year as a project engineer in Heilongjiang Mine Design Institute. He attended China Institute of Mining and Technology from 1980 to 1983, where he received a M.S. degree in Mining Engineering. Then He worked as a research engineer for Coal Technology and Economy Research Institute, China Central Mining Research Institute for three years. In 1986, Mr. Wu came to Virginia Tech to study for his Ph.D. in MIning Engineering.

Xing Wu